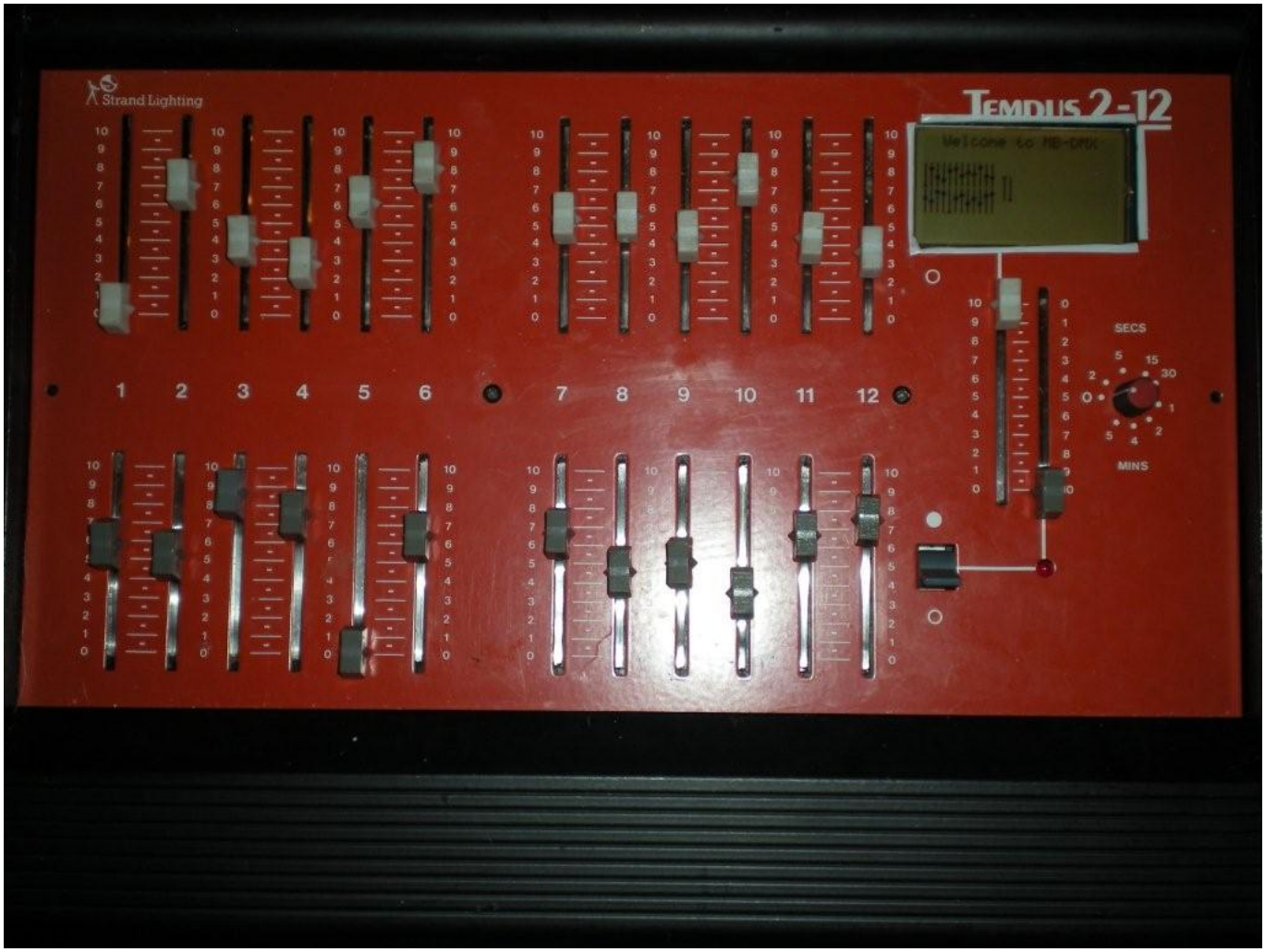
Intro

Midibox DMX is a DMX512 compliant controller for Midibox created by Phil Taylor - phil@taylor.org.uk

Originally Discussed here

Here is a picture of my test system, It is enclosed in an old Strand Tempus 2-12 case (I bought 4 very cheap!).



Concept

Over the years, I have used many professional lighting desks which apart from their very high costs, are very unfriendly to use. I may discover during the development why this is the case? I have also used cheap DJ style lighting desks which are basically boxes with a few faders and very little intelligence.

I hope to develop something that while not as complicated as the pro desks will give as many of their features as possible while not costing any more than the DJ style desks.

Expected Features

- Uses a modified MBHP_IIC_MIDI to send the actual DMX (less load on Core)
- Includes the ability to change delay times to handle incompatibilities with other DMX equipment
- Bank selectable faders (probably 8) I would like these to be motorised if I can find some.
- programmable submasters, scenes and chases (probably on bankstick)
- Due to memory limit on P16f88, maximum channels is 352 (not bad as total RAM is 368). Now reduced to 256.
- Support for ANSI/ESTA E1.20 **R**emote **D**evice **M**anagement *FUTURE*

And hopefully many more when I think of them.

Requested Features

- Use a slave core instead of MBHC_IIC_MIDI to allow full Universe of 512 channels.

Please add any other features you think would be 'cool'

Status

So far I have created code that replaces the firmware on an MBHP_IIC_MIDI module and this is currently working on 2 test DMX fixtures. I have designed a very simple TTL↔RS485 circuit using a SN75176 differential bus transceiver. This also connects to RA2 which allows control of signal direction (currently not used but should allow the addition of ANSI/ESTA E1.20 (RDM) support in the future.)

I have also written a very very simple Midi→DMX controller which reacts CC Messages, it also converts the 7bit midi velocity to 8bit. This is where I will be focusing most of my efforts for now and will hopefully become the 'actual' Controller/Desk.

I have made a couple of (very very very alpha) releases of the software for both parts and will continue to do so for as long as I can!

I have made quite a lot of changes to the IIC_DMX code over the last week or so. One serious problem that I found was that when I was sending the DMX streams at full speed (without any delay between them) occasionally they would all set level to zero! (I think the 16f88 may have been resetting itself) I searched through the code and I couldn't find what could be causing it but now I am only sending 40-50 DMX packets a second (with a timer) this problem seems to have disappeared!

I have been looking further into RDM support and it seems more than possible to achieve, the main problem at the moment is by providing 352 channels there just isn't any space for other temporary storage and the other 16 registers are all in use so I may end up reducing the total channel capacity to make way for other features. (maybe as a compile time option)

I have decided that after struggling with the limited memory of the PIC16F88, 352 DMX channels was just too ambitious. I have now reduced it to 256 channels. One of the main reasons for this decision is that I was missing too much IIC data and causing too many retries. I have now made the IIC handing interrupt based and buffered which seems to make it much more stable (I will be posting this update

once I have tested it thoroughly.)

ANSI E1.20-2006 RDM Support

I have done some of the groundwork to providing RDM support, I have a copy of the ANSI E1.20-2006 standards document and the example C source from Wybron which give full details for creating an RDM responder.

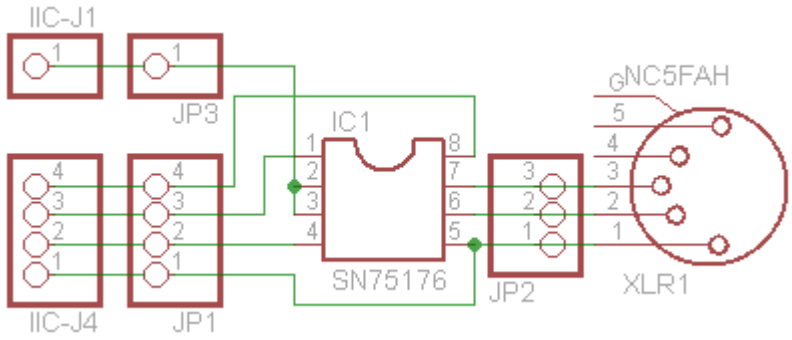
I have also been assigned a Manufacturer ID from ESTA. The reason that manufacturer ID's are important is that much like Ethernet, every RDM device must have a UID which is unique. The manufacturer ID that I have been assigned is 4D42h As it needs a 'company' name, it is assigned to my company (LAN Systems) as follows: “LAN Systems – midibox project”. Before anybody asks, this DOES NOT imply any commercial or other relationship between LAN Systems and the midibox project (there is none), they simply needed a company name to register it to.

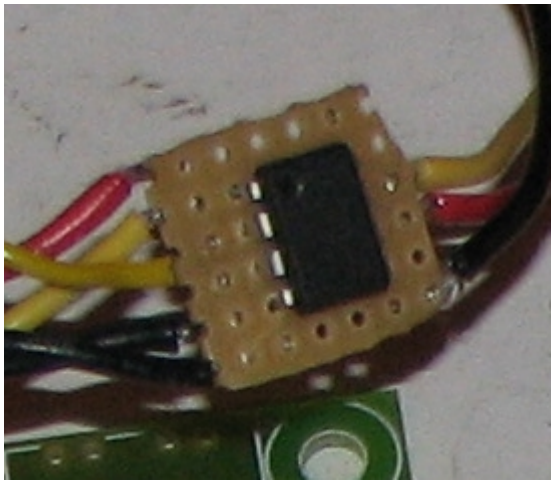
ESTA were concerned about controlling the allocation of UID's and I have committed to creating a website where people can request allocations, I will create this soon but as RDM support is quite a long way away I don't thinkt there is much of a rush!

Hardware

Pretty much all of the hardware is based around standard MBHP modules, the only addition is my simple TTL Serial ↔ RS485 interface the schematic of which is below. I made it on a 6×6 piece of strip (vero) board (I didn't bother with header connectors)

After reading the RDM spec, I realise that a simple BIAS circuit will also be required, this is only 2 extra resistors (+ 120ohm terminator) so I will post a modified circuit soon.





Firmware/Software

The software will be changing rapidly, it isn't really useful for anything other than playing with at the moment. I have not done make dist on either of them yet as they aren't really ready for 'prime time' so the mios-base kit is required to re-compile. It really is a moving target so keep checking back to see if I have updated it!

Currently the software (firmware) is available in 2 parts, there is the IIC_DMX firmware which must be burnt to a PIC16F88 (you will require a PIC Burner for this, I use the MBHP PIC Burner from Mike as it also has an 18 pin socket. I am currently throttling the DMX output to approx 30 packets a second, this allows me to see the flickering LED so I know that it is working! It seems quite stable now I am throttling it as at full speed it seemed to cause random reboots of the PIC!

IIC_DMX Firmware The IIC_DMX firmware expects IIC packets in the following format (all unsigned char), <COMMAND><CHANNEL><VALUE><TERMINATOR>.

COMMAND Is used to tell the device what type of data is being sent, the only one that I am currently using is 0xbx (like MIDI CC messages). 0xb0 selects the first bank of channels (0-255) 0xb1 will select the upper bank of channels (256-351). I may change this to be more in line with MIDI (7 bit). Once a full packet is processed, it will send the packet back over the IIC bus as confirmation.

CHANNEL Is the DMX channel and can be between 0x00 and 0xff, if a non-existant channel is selected (351-511) this will be silently ignored.

VALUE Is the actual channel value and can be between 0x00 and 0xff.

TERMINATOR Is always 0x00. I added this as I found that on start-up sometimes invalid IIC messages can be received which seem to cause the IIC_DMX problems. I will probably make this a simple checksum eventually as invalid data is a very bad thing with DMX, you really don't want random lights going on/off during your production of Phantom……..

I am also working on a method of streaming a whole bank which will be something like <COMMAND><START CHAN><END CHAN><TERMINATOR><VALUES * (END-START)><TERMINATOR> . In addition to this I will have a reverse stream which asks the IIC_DMX to return what it thinks the values should be.

The package for IIC_DMX is available here: IIC_DMX Package Currently Version 0.003b updated 08th

December 2008. This now seems pretty stable. It uses a 16 byte ring buffer for received IIC data and a CRC8 byte as a data terminator.

DMX_CONTROLLER Firmware This is a VERY simple DMX controller which I originally created to test the IIC_DMX but I hope eventually to be able to make this into a fully featured DMX controller. I am playing with 'nice' display features like wipes etc as I want the display to be 'commercial standard' (whatever that means!)

It currently accepts MIDI CC commands and converts the value to 8 bit and sends them to the IIC_DMX. Any packets received over the IIC bus are forwarded directly back over MIDI. I have also included basic support for faders via an AIN module, currently only 8 are used but more will be added when I get round to it and these just change the first 8 DMX channels. Eventually I will build a DIN module and the faders will be able to operate on different 'banks'.

The package for DMX_Controller is available here: DMX_Controller Package Currently Version 0.003b updated 08th December 2008. **Please note** Neither of these firmwares are any where near complete and may not work AT ALL. They are only really useful to somebody with a reasonable knowledge of C and/or Assembler and obviously I offer no warranty so if you blow-up your CORE/IIC_MIDI or both don't blame me!

TODO

Pretty much everything at the moment,

From:
http://wiki.midibox.org/ - **MIDIbox**

Permanent link:
http://wiki.midibox.org/doku.php?id=midiboxdmx&rev=1320872990

Last update: **2011/11/09 21:09**