This page is under construction, and I suspect that it will never reach completion, but I'll try to get it the closest possible to completion.

August 2009 update: I'm currently implementing (i.e. making) the project. The final design is very close to what I'm describing here, but not entirely. When I finish it and if the results are good, I'll try to do a global update.

About me

Okay, so my nickname is Wicked Blade. That's a bit lame but that's what I came up with one day that I needed something vaguely warrior-like to play Unreal Tournament. But this is irrelevant.

So. I'm 30-ish and what brought me to the MIDIbox community is one of my favourite passtimes: playing the guitar. I'm a rock/hard rock electric guitar player and I was looking for a way to interface my gear together. Since I've done a couple DIY stompboxes, going the DIY way was an option. And since I couldn't find anything that does what I need, and the things that somewhat do it are very expensive, I thought, why not do it myself, and here we are.

So what are my needs? Well, you have to know what gear I have to understand that. Aside from the guitars, I'm equipped with:

- an ADA MP-1 MIDI guitar preamps (an 80's vacuum tube unit that was one of the first to offer MIDI capabilities)
- a Rocktron Replifex multieffect, with MIDI capabilities, placed between the preamp and the poweramp.
- a few stompboxes that go between the guitar and the preamp

So I need something to switch presets for my MP-1 and Replifex, and also to activate/bypass my stompboxes without having to learn tap-dancing.

That's were I discovered MIDIbox, and most particularly the Pedal Box / Pedal Board projects. These are neat, and very close to what I want, but not quite it, so here is another take on the pedal board / fx looper project.

One final thing about me: I work as a software development engineer, so I'm really not afraid of the programming part, I eat C/C++ at breakfast. I'm much more of a noob on the electronics side, but I'm learning!

My pedal board/looper project

I guess I need a better name for it... I've always found the Street Fighter game series naming amusing, so I think I will go with “Pedalboard EX+ alpha”. Well, why not? So, PBEX+a for short.

This is the link to the forum thread I've started on the subject: Yet another pedalboard variation

Inspirations

Here are links to sites and pages that I'm borrowing ideas from:

- The pedal_box project
- flo's user page
- Frank Clarke's Switch Witch project
- Thomas Scarff's page on MIDI PROGRAM CHANGE and related projects

Other sources of inspiration, though not directly linked to loopers and pedalboards:

- AMZ's talk of the superbuffer, of which I will do a version with only 1 opamp
- Geofex's article on CMOS switching and other stuff

Pedalboards that resemble what I'm doing here:

- Fractal Audio Systems MFC101
- Providence PEC-2

Major design changes

Since the beginning of the project, a few things have changed, listed here:

- switched from aluminum to steel chassis (hopefully sturdier, if possibly harder to drill)
- went from 10 footswitches to 12 (2 rows of 6) – this means going from 2 to 4 custom controls. Heck, I have the space, so why not?
- channel switching, originally CMOS based (not working!) now uses relays
- went from 8 to 10 effect loops, it will be stuffy but I think I can get it all to fit and still be particable(and be able to physically separate the signal out from the digital circuits)
- went from 2 to 4 custom control footswitches

User needs

What I want is a pedalboard that handles banks of 5 patches, plus an additionnal tap tempo footswitch, and 4 additional custom control footswitches (more on that later).

So I need a total of 12 footswitches:

- 2 for bank up/down
- 5 for patch selection inside a bank
- 4 for custom controls
- 1 for tap tempo

A draft for the footswitches placement will come soon.

Additionnaly I will need an interface for edition. I think 2 buttons (edit/cancel and store) and 2 rotary encoders should do it.

The pedalboard will send program change commands to MIDI devices, and will activate/disable loops. The loops are of the most basic types: the order is fixed, so I need only a send/return pair of jack for each loop. I don't need the more complicated loops found on some commercial hardware. I'm aiming for 8 or 10 loops, depending on the room I'll have on the chassis. Aside from the loops, the unit will also provide an input and an output buffer (based on Jack Orman's), which can be disabled/enabled at will just like the loops.

The board will also feature a channel switching input jack, to command non-MIDI amps, and 2 inputs for expression pedals. The channel will be selectable on a per-patch basis. Likewise, the control assigned to the expression pedals will be customizable per-patch.

Custom controls (Cx)

So how do the custom controls footswitches work? Basically, for any patch, you can specify that the custom controls will toggle the state of any of the loops, and you can specify MIDI controller change commands to be sent when entering and leaving the custom control state (you might want to specify different parameters upon entering and leaving, so there is room for that). You cannot, however, send program change commands. The Cx would typically be used to enable/disable effects on my Replifex.

I like this idea, because it offers the same kind of flexibility as having a switch for each and every loop, but in a smaller package. I doubt that I would need more than 2 custom controls. I will probably go up to 4 custom controls, since I have some room left on the chassis for 2 more footswitches, but I don't think they will be that necessary.

Edition mode description

To do.

Chassis

I originally wanted to go with built-in expression pedals, but couldn't find a ready-made box that was big enough. So I decided to go with only the footswitches and input jacks to plug commercial expression pedals. Looking at commercial pedalboards (e.g. Rocktron All Access) and my own Zoom GFX-8, it looks like the footswitches have to be about 70mm apart to accomodate for regular foots.

The box I chose is the Hammond 1441-26 (with cover 1431-26). It is a gray-painted steel box, 1mm thick, and its dimensions are 406x203x51mm. I'm hoping it will be sturdy enough... There's also a similar offer, made of aluminum, the 1444-26/1434-26, but I'm worried that it won't be as sturdy (though easier to drill).

With this box, I can fit 2 rows of 6 footswitches. The first and last fswitches in the row are 28mm away from the edge, and the footswitches are 70mm apart. I'll try to post a drawing sometime.

Also, I need to think about the space needed to fit the female jack connectors. Measures on both my MP-1 and my GFX-8 show that you need an area of about 20x20mm to fit a female jack connector. That means that I won't be able to fit all 20 of my connectors in a row on the backside of the unit, not if I want to keep some space to fit the other connectors (power supply, MIDI connectors, etc). Fortunately, the unit is deep enough (51mm) that I can fit 2 female jack connectors vertically! Having

them paired vertically (send on top of return) isn't very nice and not easy to handle, I think, so I'm going to put them in a quincux disposition, with the input on top, and the output on the bottom, slightly displaced on the side (say, 5mm). All in all, this means I can fit 10 loops in a 205mm wide area (plus keeping a 5mm or margin from the edge). Much better than the initial 400mm! Also, the quincux disposition may help keep the box sturdy. I'll put the main input jack connector on the left side of the unit, this saves a bit more space on the backside and seems more a more logical place to me.

All in all, I still have about 200mm (406-205) to fit the remaining connectors (from right to left, when looking from behind):

- main output jack ~20mm
- MIDI OUT and IN, on over the other in quincunx, ~40mm (a DIN connector with screws is 30x20mm)
- channel switch jack (stereo jack) ~20mm
- DC connector ~20mm
- 2 expression pedal jacks (stereo jack) (in quincux too, as this avoids trouble when not quite knowing in which order to put the pedals) ~30mm

This takes about 130mm.

I'll have to be careful with my measurements, but this should work.

Possible relays

All relays are DPDT, non-latching types. Also, I'm only considering 5VDC types because that's what the MIDIBox CORE module is able to provide through the 7805 power regulator. Choices:

- Finder 30.22S, 5VDC, 125R (that means 200mW), that is what Banzai Music and Musikding offer
- NEC EA2-5NJ, 5VDC, 140mW (what Frank Clarke used, apparently)
- NEC ED2-5NJ, 5VDC, 50mW, same stuff, but much less power-hungry (Geofex apparently uses the 12V model, ED2-12NJ)

The NEC ED2-5NJ looks promising because it's power effective, and since I need to power 12 relays (possibly all at once) I may find myself with power supply problems. My only concern is whether it is suitable for audio needs (mainly noise problems), but I'm thinking this should be okay, because the voltage is DC and the intensity is low, which means that there shouldn't be much electromagnetic noise, if at all.

I'll feed the relays according to this document found on this very site (thanks to flo).

NOTE: I think I read somewhere that there was a way to avoid the clicking sound when a relay is switched on, but I can't remember how, nor where I read it. I'll have to investigate… Ah, found it. It was on Geofex, but this can't apply here because it was on a schematics using single transistors, so it doesn't work as is with darlington pairs. Maybe the darlington pairs are somehow designed to handle this too? I'll see how that works when the project will be finished…

The channel switching relays should be simple 5VDC SPST (normally open) relays. I found the Omron G6L-1P-DC5 to be a likely candidate. The coil current is 36mA (much higher than the NEC ED2-5NJ, BTW) and it looks able to withstand the 12V signal that, say, Mesa Boogie uses. I'm going to use it for Fender amp switching, though, so that's not a concern for now.

I had a hell of a time trying to understand how to wire the relays to the ULN2803. Some schematics seemed to imply that I should wire a diode parallel to the relay to avoid backlashes to the transistors in the array when a relay is suddenly switched off. After studying the matter for a very long time (along with the fact that some schematics had said diode wired backwards), I found out that this is exactly what the “common free wheeling diodes” pin on the ULN2803 is for (this is how the STM datasheet for the ULN2803 labels the pin). This explains why on the Relay example from MIDIO128, pin 10 (COM, the aforementioned “common free wheeling diodes” pin) is wired to V+!

MBHP modules

Based on the features of my box, I've determined that I will need:

- 1 CORE module (obviously), a PIC core should suffice. I will go with a 18F4620, it is cheap enough and I probably won't need all the memory it provides
- 1 DINx4 module, to handle the footswitches and the edition buttons/rotary encoders
- 2 DOUTx4 modules to handle the relays, the status LEDs and the 7-segment digits LEDs. 1 module is not enough, but I probably won't need to fully populate the 2nd DOUTx4, but it's nice to have the option of more DOUTs
- 1 bankstick, of which I will probably need only 2 EPROM chips (see next section)…

Here is the detail of what is going to be connected to the DIN/DOUT.

Digital inputs:

- 12 footswitches (bank up/down, patches A..D, tap tempo, custom controls C1-C4)
- 2 rotary encoders for edition (parameter select, parameter value change), that means 2×2 inputs
- 2 buttons (edit/cancel button and store button)
- possibly 2 inputs to know if expression pedals are plugged

This amounts to a maximum total of 20 digital inputs max.

Digital outputs:

- 12 relays for loops and buffers activation
- 2 relays for channel switching operation
- 2 7-segments LED digits, 8 inputs per digit, so that's 2×8 outputs
- 5 status LEDs showing which patch is activated
- 5 status LEDs for custom controls and tap tempo

This makes for a total of 40 outputs, so I need one DOUTx4 and one DOUTx1.

Please note that there will also be status LEDs showing which loops/buffers are activated, but these will be wired in parallel with the relays, so they don't need more outputs. But I may use specific outputs for these anyway, because I may use separate output registers for the relays, which may help reduce noise. In that case we would have:

- 12 relays for loops and buffers activation, on 2 separate shift registers, so that's actually consuming 16 outputs
- 12 status LEDs for the loops/buffers
- 2 relays for channel switching operation (make that 4, because I will need to use to darlington

arrays of 8 entries each, so let's go to the full 16 relays count)
- 2 7-segments LED digits, 8 inputs per digit, so that's 2×8 outputs
- 5 status LEDs showing which patch is activated
- 5 status LEDs for custom controls and tap tempo

That makes for 52 outputs rounded to 32+24, so a DOUTx4 and a DOUTx3.

Memory

Even being conservative, it appears that I can easily get away with 32 bytes per custom control, and then 256 bytes per patch is more than enough. I'm planning for 16 banks of 5 patches, so that would mean 5*256+256 bytes (1536 bytes, including 256 bytes for bank data) per bank (more than enough), and in turn 24576 bytes for the whole mess, that is 24kb if you prefer. Even if I got up to 32 banks, that would mean 48kb, to which I have to add a couple more bytes for the global parameters.

All in all, this shows that one 24LC512 chip of 64k will be more than enough. Mouser proposes this with part number 579-24LC512-I/P, that's perfect. The 579-24LC512-E/P works to, but I don't think I need its superior working temperature max (125°C instead of 85°C). It can't hurt, though.

Power supply

Based on the parts_faq, I think that the MPHP modules will, at worst, consume about 640mA (at 5V). To this, I have to add the consumption of the various LEDs (including the 7-segment digits) and, most of all, the relays.

The relays are the most power hungry things of the lot. And I need 12 of them! Now, if I use the NEC ED2-5NJ, which consume 50mW, that makes for a total of 600mW (and these are low-current types!). Translated in current, that means that the relays will consume 120mA if all of them are activated at once. As a reference point, this value would go up to 336mA with the EA2-5NJ, and 480mA with the Finder 30.22S.

So, the modules and the relays can amount to a 760mA current draw (1120mA with the Finder relays! that's really high!). I think that the LED consumption is negligible in regards to that, but I'll have to check...

But this is the maximum continuous current draw, we have to take the pikes into account too. I've been told that aiming at 2A is a good choice, so I'll try to do that.

As far as voltage is concerned, the PIC and associated ICs need 5V, the relays need 5V too, but for the input/output buffers opamp, I need to get as much headroom as possible. 9-10V might do the trick. Given that voltage regulator tend to drop about 2V from their input voltage, I've decided to go with a 12V power supply. There are plenty of 12V 2A power supplies available, but they tend to be switching PSU, which we do not want, because they often add unwanted noise. I've found a linear 12V 1.5A PSU instead at Thomann, the THOMANN NT 1215 C unit, which is a replacement for Casio products. It is indeed linear (I've ordered it blindly, but it turned out good), the only caveat is that it has a plug that is not the “usual” DC 2.1mm jack, but it's easily swapped for anything you like.

It appears to me that I need to manage 2 separate grounds for analog signal and digital signal. I don't quite know how to do that. I've been told I could put some kind of low pass filter somewhere, but I

don't quite know how and where. I believe this should go between the digital and analog grounds, so that the analog ground gets a filtered version of the digital ground, but how should I go about it? I got no answer about that on the forum, so I've asked around on other forums. I decided to simply duplicate the voltage filtering/regulating circuit from the CORE, and this one will feed the relays and my opamp. I'll be careful to use star grounding, and I hope that this will be enough to keep noise to an acceptable level.

Parts list

In progress.

Aside from the MBHP modules, I will need:

- the box (body, plate, and screws)
- 12 SPST non-latching footswitches
- 2 SPST non-latching buttons (for edition)
- 2 rotary encoders, and 2 knobs – Bi Technologies offers the EN12-HN22AF18 for instance, which seems fitting
- 2 7-segment LED digits, 0.8“, common anode, blue (I like blue, and I'm tired of the usual red stuff)
- 1 16×2 LCD display, white on blue
- 10 5mm LEDs for patch/custom control/tap tempo status (5 blue + 5 green, or 10 blue, or something in between)
- 10 5mm inwards reflecting plastic LED mouting clips
- 10+2 3mm status LEDs for loop/buffer status (10 blue, 2 green)
- 12 3mm LED plastic mouting clips
- 36 100ohms resistance for the blue LEDs and digits (they're usually 3.2-3.3V/20-30mA, this would drive them at 17-18mA, another option is a 91ohms resistor)
- 12 NEC ED2-5NJ relays (DPDT, for loop activation)
- 2 Omron G6L-1P-DC5 relays (SPST-NO, for channel switching)
- 2 ULN2803 for relay driving (can handle 16 relays), with sockets
- 1 7805 power regulator and associated heat sink for relay power supply - see below too
- a few breadboards
- 10 switched mono input jacks for loop returns - e.g. Switchcraft 12A (offers a way to connect to ground or input signal if no jack is inserted) or Neutrik NYS234 (stereo, smaller, cheaper, encased, but has some straight pins)
- 10 non-switched mono input jacks for loop sends - e.g. Neutrik NYS229, but the NYS234 is cheaper, so why bother?
- 2 non-switched mono input jacks for main input and output
- 2 switched stereo input jacks for expression pedals
- 1 non-switched stereo input jack for channel switching, note that this one needs to be insulated somehow from the chassis, as the signal for channel switching won't necessary share a common ground with the audio signal and associated switches
- 2 5-pin (or 7-pin) female DIN sockets, chassis mounted (for MIDI IN/OUT) – included in Mike's Midishop kit
- 1 2.1mm DC jack for power supply input, insulated
- 2 OPA134PA opamps (or 1 OPA2134PA) and sockets for input/output buffers, with associated resistors and capacitors - see below
- 2 24LC512-I/P for the bankstick
- Some PCB standoffs

- Stuff for the channel switching system (taking a look at the Fender schematics)
- a few breadboards
- ...

Notes:

Blue LEDs have different working conditions than regular red or green LEDs, so I'll have to adapt the resistor values. Most often they have a 3.2/3.3V forward voltage and work at 20 to 30mA. I'll go with 100 ohms resistors, which should feed the LEDs with 18 to 20 ohm (depending on the voltage) and should be enough.

I'm not sure as far as the power regulator is concerned. It appears that the 7805 has a 1A output current, and I may need more than that at pikes. but is this 1A value a 'normal' value and the 7805 can handle higher pikes, or should I modify the core and use a L78S05CV by STM instead, for instance (the L78S05CV outputs a 2A current)? To get an additionnal power supply line for the relays/opamps, I'll just copy the circuit from the CORE module, so I'll need:

- a bridge rectifier (W08G-E4/51, like SmashTV uses, it handles a higher current than the B40C800 specified on the wiki)
- 1 2200uF electrolytic cap
- 1 10uF electrolytic cap
- 1 330 nF ceramic cap
- 1 100 nF ceramic cap
- a linear regulator (LM7805 or L78S05)

For the input and output buffers, I chose the OPA2134 for several reasons. First, it's a Burr Brown, and they are quite known for doing very audiophile stuff. I've swapped many opamps in my MP-1 with OPA2604 opamps, with great success. But here I chose the OPA2134 over the OPA2604 because it is, apparently, more transparent than the 2604, which is more tube-like sounding. All this is hearsay from the internet, of course, but I have to make a choice at some point. Anyway, I don't really want the buffers to be part of the sound coloration, here, so I'm trying these.

Here's the partlist for one buffer:

- 1x OPA2134PA (half of it actually)
- 1x DIP8 socket
- 3x 100k metal film resistors
- 3x 2.2M metal film resistors
- 2x 0.1µF metal film capacitors
- 2x 10µF electrolytic capacitors

Note that I have to double the quantities because I want 2 buffers (except for the opamp which is actually a dual opamp). Again, more information can be found on Jack Orman's AMZ site (look for the “Basic Buffers” and “Super Buffer” pages).

For channel switching I have considered using a CD4053 like geofex does, but it needs a -5V input besides the +5V and ground. And I don't have that. It doesn't look like it handles -2.5/0/2.5V well, so I have to go another route. Here's a topic I opened on the forum on this matter: Connecting a HC4053 to DOUT Another option is the “solid state relays”, which are described in this page... But in absence of any advice on the subject, I will simply turn to good old SPST relays, like the Omron G5V-1-DC5 (which is actually an SPDT relay but us cheaper thant its SPST counterparts!).

From the Fender Hot Rod Deluxe schematics, I gather that to drive the “drive” and “more drive”

channel switches, aside from the relays I will need:

- 2 1N1448 diodes
- 2 47ohms 1/4W 5% resistors
- 2 22uF/25V electrolytic caps

The schematics can be found easily on the net. Amps differ widely as far as channel switching is concerned, so you'd better check the schematics to match your amp! A simple way of being easily compatible would be to simply wire the channel switching stereo input jack to the relays (one relay for the tip, and one for the ring) and make an external box to translate correctly your footswitch signal to that.

Loops wiring

Since I'm getting stereo switched jacks for my loops sends/returns, I thought of the following wiring:

- send jack has T wired to signal in (relay), RN wired to signal in too
- return jack has T wired to signal out and TN wired to R of the send jack.

This way, it will handle most pluging situations:

- if send is plugged and return is plugged, that gives signal in and signal out
- if send is plugged and return isn't, then return T is wired to send R, which is actually ground (since we use mono effects), and the overall effect is silence, but the effect plugged to send will actually receive the signal (useful for tuners, for instance)
- if send is unplugged and return is plugged, then signal in is ignored
- if none are plugged, then signal simply goes from send RN/R to return TN/T, which means that the relay state doesn't matter.

Building process, things learned and things I would have liked to be aware of beforehand

(in progress)

- The SDCC compiler doesn't support a variable implementation in .c with 'extern' definition in .h nicely. Makes it difficult to design a nice architecture.
- PIC limitations mean that you can't declare variables that use more than 256 bytes in RAM. This is the cause of the unintelligible linker error message “no target memory available for <something>”. I had a big struct that I just split into 3 smaller ones, now it's ok. I've looked for an explanation for a long time and only found it in a passing comment on another forum thanks to Google…
- Compiling for the 18f4620 makes the linker output warnings about unrecognized PIC and using the generic PIC18cxx type. The warnings are apparently considered ok to ignore. I personally don't think warnings are ok in any from (and I fixed them, search the forum), but hey, I'm not coding the gputils or SDCC…
- Code::Blocks doesn't filter compilation warnings and errors very well, so it's better to check the build log tab, or you might entirely miss a message, or read only a partial message, when the full description would help.

- It's often better to use a lot of *if* statements than *switch* statements, because the latter have a non-negligible overhead. Also, it is often much more efficient (memory-wise as well as performance wise) to use arrays where applicable.
- Because of the banked structure of memory in the PIC, you should take special care where you put your data to avoid unnecessary BANKSEL instructions (these are the one that select a particular bank in assembly language). Similarly, I found that it's often better to work on a local copy (into a local variable) of a volatile global variable, than using it repeatedly: the compiler has no way of knowing if an interrupt routine has modified it and will read it back too often, including using BANKSEL instructions. Remember that declaring a local variable doesn't necessarily mean that there will be some space in RAM allocated for it, it could also be simple registers (it's often the case!).
- The optocoupler in the CORE module (a 6N138) is a fragile thing, so buy a few spares, just in case
- For use as a computer MIDI interface, you really need either a gameport MIDI cable (if you motherboard has a gameport) or a USB MIDI cable. I now know for a fact that my Line6 PODxt and my Boss DR-880 can't be trusted, even though they sport USB and MIDI outputs. The PODxt filters out SysEx messages and I couldn't get my DR-880 to forward anything.
- Noise! It turns out that having the DOUT modules chained is a recipe for noise. Specifically 1kHz noise (if that's the frequency to which you set the shift registers refresh). After a hard time investigating, this gets fixed by star-grounding the grounds of each module to the ground pin of the big cap connected to the voltage regulator on the CORE module.
- The documentation about MIOS_BANKSTICK functions is unclear about that, but it seems that if you want to use the page reading/writing functions, you MUST have aligned adresses (i.e. adresses where (addr & 63)==0), otherwise all kinds of trouble appear
- I would have dearly loved to know about the watchdog timer (WDT) before, that would have helped me understand why my app kept rebooting while it was trying to initialize the banksticks! Days on debugging have been wasted on this :(