MIDIbox Master Keys

The idea originally came from this thread, to enable a broken master keyboard that only sends on channel one, to have keyboard splits and send on other channels... Typical of me, I had to make my design for my “ultimate” master keyboard, and it's a modular kind of setup.

The code so far is a mutated MB Router, and the idea is that incoming messages are modified byte-by-byte basis (nibble-by nibble for status bytes) according to a set of rules which the user defines. The rules can modify the bytes however you want them to (although rules should be written with latency in mind), and a number of variables are available for the rules to use in modifying the bytes.

There are a few variables within the individual rules, which can be used by those rules alone, and also a larger pool of variables which are available to all of the rules. Most of these 'global' variables store the last midi bytes received - the last note number for each channel will take 16 of those variables, ditto for CC numbers and pitchbend, and each CC number will have it's last value stored also.

The rules work like this: For each rule, first you set the criteria for incoming messages to match this rule, by selecting a ptype (what kind of midi message it should be) and a maximum and minimum value for each of the three bytes.... so you might specify that your rule should be triggered by a note on/off message, on channel 0-0 (Ch 1), where the note is within the range of 63-127 (the top half of the board), and velocity between 0 and 127 (all of them).

Your rule can then use one of it's internal variables as a shift for the note value, let's say you set the variable to -24 (with an encoder) so you drop 2 octaves...You use another internal variable which you set to +1 to put the new keyrange on channel 2. That's a pretty normal keyboard split...

Now let's say we have another rule, setup the same way, but this time it has the third byte, which is velocity, set to be overridden by the modwheel, to allow more precise velocity entries. This variable, CC0's value, is global, not specific to this rule, so it can be used by any or all of the rules.

I'd like to make it port aware too, and have scaling and quantising of values and force to scale and save to bankstick and all kinds of things in it... Just how much it can do without effecting latency is a mystery until I finish the code though. I've had it on the backburner but it has been started... Once again the tricky part is making the LCD menu system :/ I don't want it to be confusing, it should just be a set of named options, none of 'this byte, that byte' like the technical description above, it's too confusing for most creative situations. I really need to get this whole LCD/menu thing figured out...