Introduction

This is the place to come when it comes down to my activity around MIDI and midibox.

Some random photos of my house, my workshop, projects and whatnot

About

Über-nerd from Germany, born Jan 1982. Studying computer science, had professional training in high-frequency analog electronics, audio signal processing and digital circuitry. Heavy EAGLE user. Programmer since the age of 8, starting on Commodore 64. Languages include C/C++, C#, Java, Basic, Python, Perl, PHP, assembly (8085,8086,PIC16, PIC18,AVR), SQL and many more. Beware! I'm an Apple user, but have a comfortable foot in all three worlds (Mac OS, Windows, Linux).

I've been actively involved in the midibox scene since 2007, but had been watching the project since around 2005 before. Since I grew up with the Commodore 64, naturally the SID was my entry point :)

Music side, I'm old fashioned, just getting into electronic music creation and exploring more and more of it every day. I'm a rather good singer, and comfortable on guitar. Besides that, I'm getting better in keyboard playing every day and can do some basic stuff on drums and bass guitar.

Music taste is broad; I love classic progressive rock, modern progressive metal, classic heavy metal, 80's rock, classic rock, blues, jazz and I do enjoy some classical music too from time to time; I came to like ambient electronic music, too, but I really hate techno music and rap as well as hip-hop (sorry guys).

Builds

MB-6582

I built an MB-6582 in 2008.

Photos are available under http://users.sr-iserlohn.de/mb6582/

Projects

Double-manual MIDI Keyboard Controller

This is the real-world MIDI project I'm in the process of building right now. It incorporates two 61-key manuals, and a lot of digital and analog controllers. This project is not based on the midibox hardware platform. It uses two ATmega32s and one ATmega2560 instead. The mega32s act as the keyboard matrix scanners and communicate with the mega2560 via USART; the mega2560 handles all the calculations and control surface I/O as well as analog signal handling plus two MIDI in/out pairs.

All controllers as freely assignable to CC, RPN and NRPN signals. Analog controllers include two joystick controllers and a ribbon controller based on flat touch potentiometers (more on this part of the circuit later).

At the moment, I am assembling the chassis, the electronics part is almost finished and programming is well underway. I expect to be done with it in spring 09. I've documented the becoming of this monster with my camera extensively so far, so expect a bunch of photos soon.

Projects in design stage / waiting queue

4x4 / 8x8 MIDI hardware router

~~This will be using basically the same hardware as the Keyboard, so this is mainly a programming task - and it's in fact the elder project of the two. First stage is developing a 4×4 hardware router using only a single microcontroller, if that works to satisfaction, I plan on expanding to using two, interconnected via SPI for rapid data transfers. The routing table will be easily set up by using a matrix very similar to the one on the mbsid. nILS suggested use of a touch screen, which I think is pretty interchangeable since there is no real difference between switches and phototransistors.~~

~~Since the introduction of the gm5, in conjunction with MIDI-Ox, this seems less urgent than ever, so I will put this on a hold.~~

16x16 programmable MIDI patch bay

This evolved from the MIDI router idea above. A whitepaper describing the device and the use cases is in preparation.

MIDI Sequencer

~~This is in early concept stage, and atm just an idea I tinker with. I like the idea and concept of the midibox sequencer, so maybe I'll decide to abandon this and build a proven solution instead... Anyway, this one is the third “project” based on the mega2560 hardware (if it's ever done). As I said, I like the concept of the MBSEQv3, but I find it quite hard to use from what I've seen and read by now, so I started doodling in aim of a more entry-level friendly UI. Update January 2009: I've worked on the Sequencer concept over the last few weeks and fleshed it out to a great deal. Currently, I'm aiming for mega128 instead of mega2560, as this one won't need too many I/Os. The hardware concept is somewhat similar to the CORE32 regarding peripherals and such, and there's a chance I might switch to STM32 after all, but I'm not sure about that right now. So far, my plan includes the use of several modules to do the job. There are six microcontrollers currently involved, all the same hardware design, but for different tasks. One is acting as a MIDI master clock, which outputs MIDI clock data on a dedicated MIDI output, plus a clock gate signal, which will be used internally for synchronization. One controller will act as a master sequencer controlling all other modules, handling the UI and doing file operations on an SD card. The other four controller modules will be the actual MIDI sending “sub”sequencers, each one playing four tracks. Several sequencer devices can be chained together to allow for (theoretically infinite) expansion; such chained seq's will not require a master clock module present, but still do require a master controller module present. Everything, from the UI, displays,~~

~~internal communication, file system access etc. will be based on SPI. Therefore, each module will have 3 SPI interfaces, each one capable of adressing 9 different SPI busses; one using the hardware SPI, two using software SPI. I chose the subsequencer-approach with several independent, gate-synchronized seq's because this offers very interesting opportunities to work with polyrhythmic sequences (i.e. one sequence consisting of 12 16th-notes, one sequence consisting of 8 16th notes, one of 3 8th notes etc.) which I find rather funny and worth exploring.~~

Update February 2009: Joined forces with stryd_one, as his vx32 Sequencer and mine were very similar in approach. As of this, target is now core32 / mios32 (plus a PC application using the same code). I'm responsible for the UI and the PC port of the app, and development is already underway. For the PC port, I use Qt4, so it's gonna be portable to Windows, OSX and Linux. Whee!

Update July 2009: The library used for GUI is now JUCE, which eases the overall work a lot, since with the introduction of the MIOSJUCE port, we can work directly with MIOS calls on the PC, too. The GUI is optimized to be used on a PC with a small touchscreen.

Research

Multitouch screen

Some random sketches exist. I'm opting for a computer multitouch screen and use that for some musical applications. (This will have to wait for a long time)

Multitouch pad

What you can do in a big screen, you can do in the small, too… This is a research project regarding the use of FTIR technology to build a small multitouch pad acting as a buttonless control surface for microcontroller circuits. For this, there is no screen involved, and no camera either, like you would use in bigger setups used for computers like the one up the list. Instead, I plan on using LEDs as indicators and phototransistors as “button” replacements which function from the FTIR (frustrated total internal reflection) technology and will allow for “buttonless integrated display and controls”. This is quite cost effectively doable, yet I still need to figure out if it works for small-scale application.

The function principle of the sensing portion is described here:
http://files.tachilab.org/intconf1900/tanie198410RVSC.pdf