Midibox Stuff by John E. Finster

Hi, my name is John. I live in Germany where I also studied Philosophy and Cultural Sciences. I became aware of TK´s ingenious Midibox Project in 2007 and started to learn how to build my own boxes. My first ones consisted of nothing more than a few pots and buttons, a display and a cardboard box. After a few financial crisis and the birth of our wonderful daughter I was forced to lay low for a few years and to take care of more important issues at hand (jobs, family, studies,…). But I dropped by occasionally, read up on all the technical stuff and followed the development of the Midibox Phenomena.

So I never gave up on my Midibox excitement and today I am delighted to present the fruits of my labor ×.

Many Thanks to TK for creating the Midibox Platform and to the community, which is one of the nicest and most helpful communities I have ever encountered.

My Midibox Projects

Here you can observe some of the projects I´ve been building.

Miditool for Finale 10 - December 2012

This was my first finished project. I built it as a christmas present for my dad. He plays the trombone and uses Finale 10 on his PC to create and edit scores. But he has a lot of trouble to enter the notes with a mouse or a computer keyboard. He doesn´t play the piano, so he couldn´t use a midi keyboard to do this, so I thought with my Midibox knowledge I could help him a little bit.

This is what came out of it. It´s not pretty, but it´s practical.

It uses a half stuffed STM32 core and one DIN module.

It´s currently running the MB_LC v2 (32bit) firmware. At that time the MB_NG firmware wasn´t finished and since all the midi events it sends are getting translated by Bome´s Midi Translator anyway before they can be used in Finale 10 I didn´t feel the need to wait.

Midibox Sequencer V4 - June 2013

The next Project on my list was a SEQv4. Since I didn´t like the "sidebar" look I went for a more compact and symetrical design.

Instead of seperate GP encoders and leds I put in some nice illuminated encoders. They can be connected just like normal encoders and leds, they have seperate pins for each of them.

Since I used a custom design I put all the buttons/leds/encoders on a standard 0,1 inch vectorboard. I had to cut a hole into it because I had to fit the data wheel in between.

Here you can see the back of the frontpanel with the lcds in place and the jog wheel case. I didn´t use any holes in the frontpanel for mounting screws, I relied solely on a two-component adhesive ("UHU Endfest 300" in my case) that´s why I had to roughen the surface. I did that even with the lcds. First I put the screws into the lcd cases, placed the lcds correctly and locked their position with a bar clamp. The screws are held up with the pegs. After that I placed a drop of adgesive under the screws and removed the pegs so that the screws could "glide" into the adhesive.

I placed some placeholders (screw with a flat head + metal tube) on the "button-side" of the pcb, put some adhesive on the frontpanel as well and gently placed the pcb on the frontpanel. The buttons and knobs guided the pcb to its right position. The leds are just placed roughly on the pcb before I glued it onto the frontpanel and soldered afterwards.

Just a quick look inside the case before I closed it up. Once without and once with the necessary cables. The case is made out of beech wood, some detailed pictures will be provided later on the site.

I left a little to less space between the pcb connectors and the lcds, so in order to connect all the modules to the pcb I had to remove the lcds again. But this went ok, the adhesive didn´t break or crumble or anything.

## Midibox NG / Mackie Control Clone - August 2013

My third project was a slightly bigger one. I´ve always wanted to have a Mackie Control based midi controller. And with the new MB_NG firmware coming up more display options were available so I decided to go with some nice SSD1306 OLED displays instead of the regular character displays.

In the end my box features:

- 9 Penny&Giles motorfaders PGFM3200 (8 track faders + 1 master fader)
- 14 encoders
- 2 jog wheels
- 70 multi-purpose buttons
- 18 SSD1306 displays
- customised fonts and icons

Since I successfully built my SEQv4 I used similar methods when I built the Mackie Clone. All the buttons/encoders/displays are soldered on a standard 0,1 inch pcb. I used a 500mmx100mm pcb and placed 14 of the displays on it. For the remaining 4 displays and the corresponding buttons I had to attach a smaller piece at the lower right side. This piece was a leftover from the SEQv4 pcb.
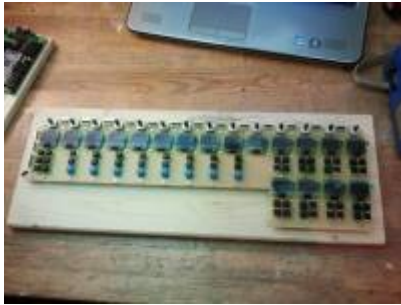
Placing and soldering the buttons and encoders wasn´t a problem. For the displays I didn´t want to span a gazillion wires so I tried another method. I used a lot of cutoff resistor legs (that´s why it is important to collect all the scrap) and layed down six tracks on the backside of the pcb for all the lines the displays share (VSS,VDD,SDA,SLK,…). The displays were connected with short wires to the tracks. I only connected the individal CS lines with an individual spanned wire. Although this approach is a clean and comfortable method to connect a lot of displays, I CAN NOT RECOMMEND this for bigger projects. I had to constantly resolder the tracks because they are very brittle and break every time the pcb gets bent just a little bit. I won´t use this method for future projects.
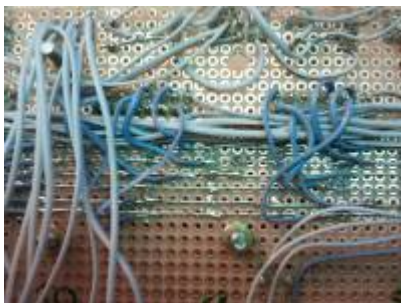
The connectors for the buttons/encoders I placed on the top edge of the pcb, like with the SEQv4 pcb, the connectors for all the displays I placed at the left edge.

After I finished soldering the pcb I cut out the spaces for the faders. I wanted to save space so I placed the track buttons (for mute/solo/select…) between the faders. This was easier said then done! The pcb material is very sturdy and it took a small drill, a pair of nippers and a pair of tongs to do so without damaging the already soldered parts.

After that I placed the pcb onto the frontpanel and glued it in with UHU Endfest, like I did with the SEQv4.
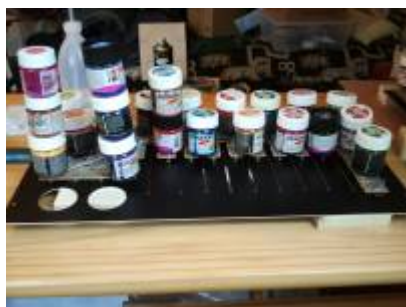
The last step was attaching the motorfaders and the jogwheel cases and putting everything into the case. Here you can see the whole box from the inside before I closed it up. (The small pcb the hammer is standing on is an unsuccesfully built midi extension for the 3. and 4. midi in/out. I left it in because I want to properly finnish it some day, but for now I don´t really need it)

The case is also made out of beech wood. I will post some pictures of how to build one in the next chapter. The small holes you can see on the bottom of the case are for the faders. I wanted the case to be as slim as possible, but I didn´t consider the cables coming out of the fadermotors ☒. I didn´t want to redo the whole case (I only have limitied access to a woodshop) so I just cut out some holes.

Building a nice wooden case

Here are some pictures of the cases I built for my SEQv4 and Mackie Clone. Both are made out of beech wood, a very dense and firm wood, which can be worked with (cut, sanded, drilled into,…) very easily if one has the right tools like a sharp (!) saw blade, an electrical sander, etc.

Because I only have limited access to a woodshop (at my parents house, I live in a small flat some distance away), I made both cases simulaneously. Both consist of 4 frame parts and a baseplate out of slim cheap pine wood.

For the frontpanel there is a small strip of wood to hold it.

The trick with the baseplate is to cut a gap along the lower bound of the frame parts.

Once all elements are finsihed the 4 fram parts are just mortised, so there are no screws necessary. The baseplate is put in the gap of the frame parts.

After that all the frame parts get glued with wood glue and helt together with 2 big bar clamps until the glue has dried.

I had a lot of help from my dad with the cases, not only has he the necessary woodshop and tools, he also has a bit more experience than I do and more ideas when it comes to wood.

Here he is acknowledging our work

Midibox Mods

Here I´d like to publish stuff related to software. I´m not a programmer so what I will be posting here is not going to be deep and ingenius software mods. Instead there is going to be a list of modifications and scripts I figured out while experimenting that I find usefull enough to share with others here.

I still think a word of caution would be appropriate:

None of the stuff in this section is part of the official firmware repository. So any use you make of it will be happening at your own risk! If your Midibox suddenly blows up or melts down while exercising the tips and tricks you find here it will be your own fault, not mine. Do NOT apply any of this stuff here if you´re not FULLY aware of what you are doing!

Frontier Alphatrack Clone - script for MB_NG

The Frontier Alphatrack was one of my favourite controllers once. I used it until it broke down. Today I´m into something a little bigger but I always thought about integrating some of the funcionality into my Midiboxes. So I created a MB_NG script that emulates almost all midi events of an original Alphatrack, the lcd display message included.

The Alphatrack´s host message is located on the right side of the screen. The other stuff on the left side (Volume bar, Solo/Mute/Rec state and "Vpot"assignement) I integrated, because I created this script with my SEQ4 and this looks more clean than using the SEQ4´s leds.

For the Alphatrack clone a 2×16 character lcd is sufficient, but I think it doesn´t hurt to have a little more lcd space for custom labels.

Instead of the fader I´m using an encoder, the volume level can be reflected on the lcd, the touch sensor is replaced with a buton event.

The original Alphatrack features touch sensitive encoders, which is a very nice and handy feature. In my design I also replaced these touch sensor events with simple button events, but with some metal-shaft encoders and metal encoder knobs one can easily build some touch sensitive encoders.

All other buttons and leds are assigned in the NGC script, just the hw_ids would have to be adapted.

The MB_NG Firmware has to be modified a little bit, inside mbng_event.c a number has to be altered:

At Line 3755:

```
u8 x_wrap = (*stream == MBNG_EVENT_SYSEX_VAR_TXT56) ? 56 : 64;
```

has to be altered to:

```
u8 x_wrap = (*stream == MBNG_EVENT_SYSEX_VAR_TXT56) ? 56 : 16;
```

So that the variable ^txt can set the cursor to the second line after 16 characters.

Then the MB_NG source has to be compiled again and uploaded onto the Midibox.

I tested it with Reaper (using the genius Alphatrack-Pro extension) and with Reason 5, no problems so far.

There are some issues I couldn´t find a solution for yet:

1. Using an encoder for volume doesn´t work completey. One can send the volume value, and the midibox somehow "picks up" the right volume value from the daw, when the encoder is moved. But the Midibox doesn´t receive this value, when it is just sent by the daw.

2. I wasn´t able to recreate the touchstrip from the Alphatrack. I can create 2 buttons with the events for a 1-finger touch and a 2-finger touch, but when I tried to emulate the touchstrip movement with an encoder, it didn´t work

I believe this has to do with the pitchbend values. The encoder would need to send/receive a 10bit

pitchbend value and the touchstrip a 5bit pitchbend value. But the MB_NG only provides a 14bit pitchbend value. This is not really a problem for me, since I wouldn´t use any of those 2 features for my own box, but for a complete clone, those issues would have to be resolved.

Download the scripts here:

http://midibox.org/forums/index.php?app=core&module=attach&section=attach&attach_id=10837

inside the *.zip is a *.ngc configuration script and a *.ngl script for the lcd labels you see on the pictures above.

How to create custom GLCD fonts/icons for Midibox NG

I compiled this tutorial mainly because there aren´t a lot of informations available on this subject and I spent days to figure this out. It requires some amount of work to implement custom fonts and icons in a Midibox_NG, which I will try to guide you through as well as I can. My programming skills are very limited, so there might be better and more efficient ways to do this. So I consider this approach just one possible way.

This tutorial is presented on a Windows machine. I don´t have access to MacOS or Linux, so I couldn´t tell if all the steps are as equally performed on those systems. I´m sure all the required tools are available in a similar way.

Before I begin I´d like to point out some limitations:

First, the height of fonts and icons is limited to a multiply of 8 (so 8, 16, 24,…pixels are possible). The width doesn´t seem to have limitations like this.

Second, the main goal here is to replace regular fonts with whatever graphics you´d like. In consequence, to use those graphics you have to point to a regular letter or a digit. So if you want to use a mute icon for example and this mute icon sits on the spot where the regular "A" would sit, then the label of the element in the NGC script has to print "A", just with another font. It can get confusing sometimes. It gets easier and more clear if you use dedicated labels inside an external NGL script. I will come to that later.

Third, this tutorial only aplies to graphical lcds.

Ok, let´s get to down to business

Requirements

- GIMP (or any other graphics software that is able to export to *.xpm, Gimp is free)

- A texteditor (like Notepad++ or similar)

- Perl (I´m using ActivePerl, can be downloaded for free) installed and added to the system path.

- All the tools necessary to compile the firmware (as stated on the ucapps programming site) including the complete Midibox repository.

- a MIOS32 compatible core running the Midibox_NG firmware and (of course) a graphical display like KS0108, SSD1306,...

When you have not done so already, the Midibox_NG firmware should be compiled at least once before you start to mess with the code. On the one hand, you have asured that your system can compile the firmware without problems. That makes looking for errors much easier later . On the other hand, the Windows command window only shows the files that are updated during the compiling process. So it is much more clear to see what files are updated and if the fonts are implemented properly.

1. Creating a font image file

What we need at first is a font image that looks like this:
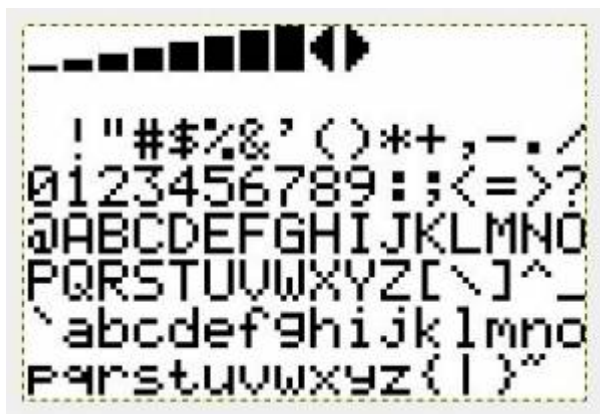
This is an original font file from TK. It consists of 8 rows and 16 columns, that makes 128 cells, so technically 128 icons can be put in. The top row is used for the default bar and some special icons used by the Midibox. The second row is empty for it´s not used by the Midibox_NG firmware (afaik). You could also manipulate the default bar and arrow icons, but I will focus on creating usable icons in the six lower rows.
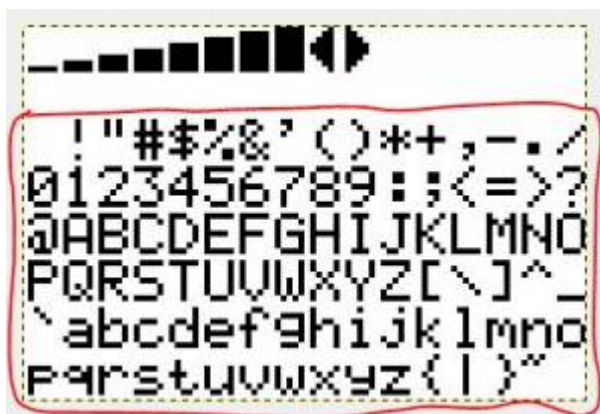
So, at first you need an empty graphic which you´re overlaying with guidelines, creating 8×16 evenly sized cells. Every cell has to have a height of a multiply of 8 pixels (8, 16, 24, ...).
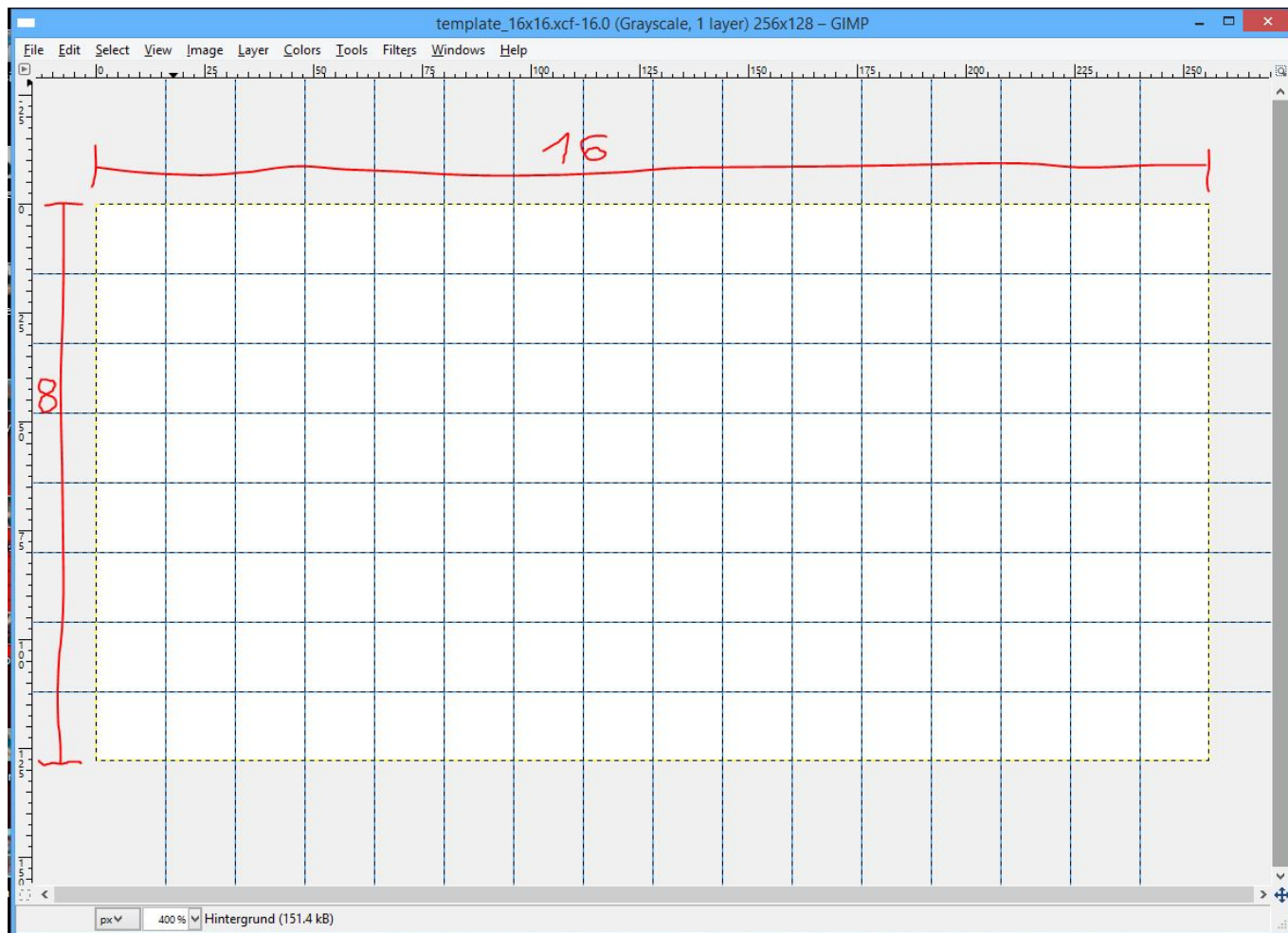
Now you can fill the cells with whatever graphics you like, but only black and white are allowed. In this example I put in a font named "crystal" I found on the web and modified it a bit. Here all the cells have a size of 12×16 (WxH) pixels.
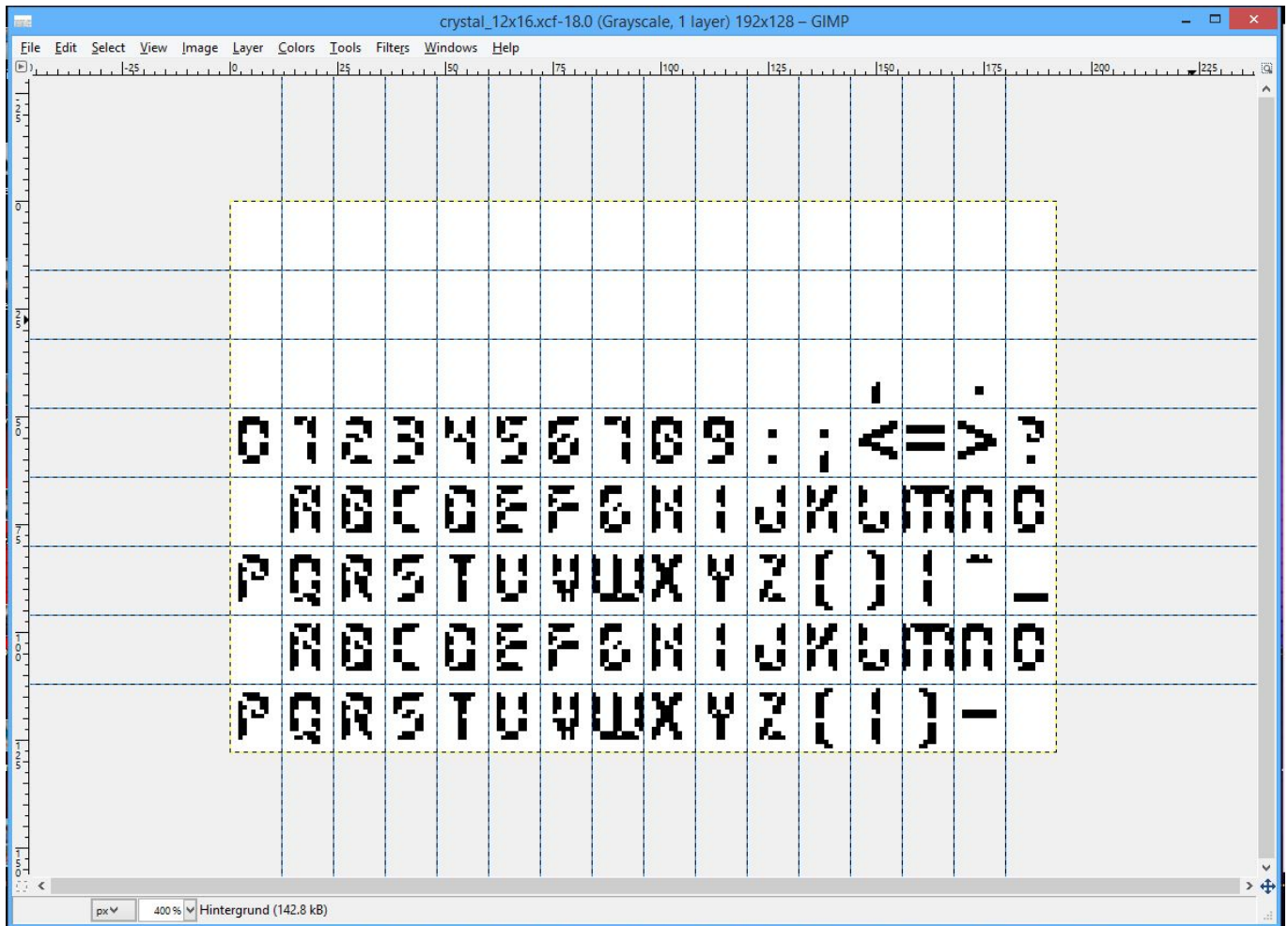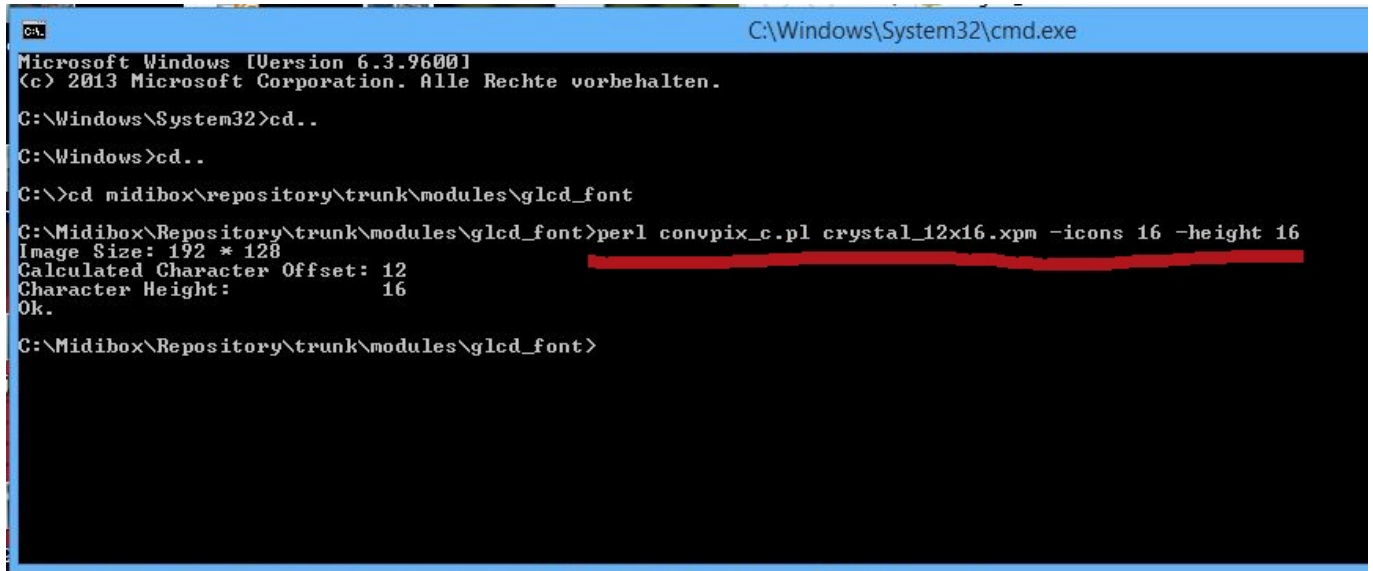
Next you have to create a *.xpm image file. Within Gimp the picture has to be exported as a *.bmp first. Then the *.bmp has to be opened in Gimp again and then (finally) exported as *.xpm. (I don´t know why it has to be a bmp first, but I tried it without this additional step and the xpm was not compatible.)

Now you have a *.xpm file ("crystal_12x16.xpm" in my case) and can proceed.

2. Converting and preparing the image file for MIOS32

For the next step you have to move the image *.xpm file into the MIOS32 repository (…\trunk\modules \glcd_font). Inside this folder there are the other Midibox fonts. Also there is a perl script called "convpix_c.pl". You have to use it to convert the *xpm file to a MIOS32 compatible font file. To do that you open a dos window (cmd.exe), change the directory to the glcd_font folder and execute the perl script as shown in the screenshot.

Next to the script name and the name of the file that is to be converted there has to be a parameter "-icons XX" that gives the number of icon columns and the parameter "-height XX" that gives the height of a single icon. The script then calculates the width of the icons.

What we now have is an *.inc file of the image which we´re going to change to a *.c file simply by renaming it. Windows will complain eventually, but we can ignore that.
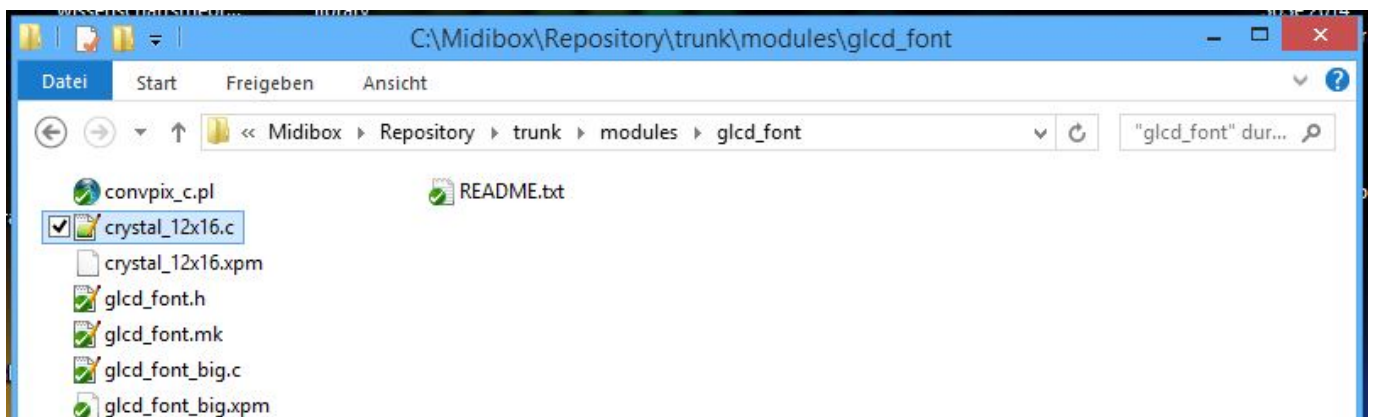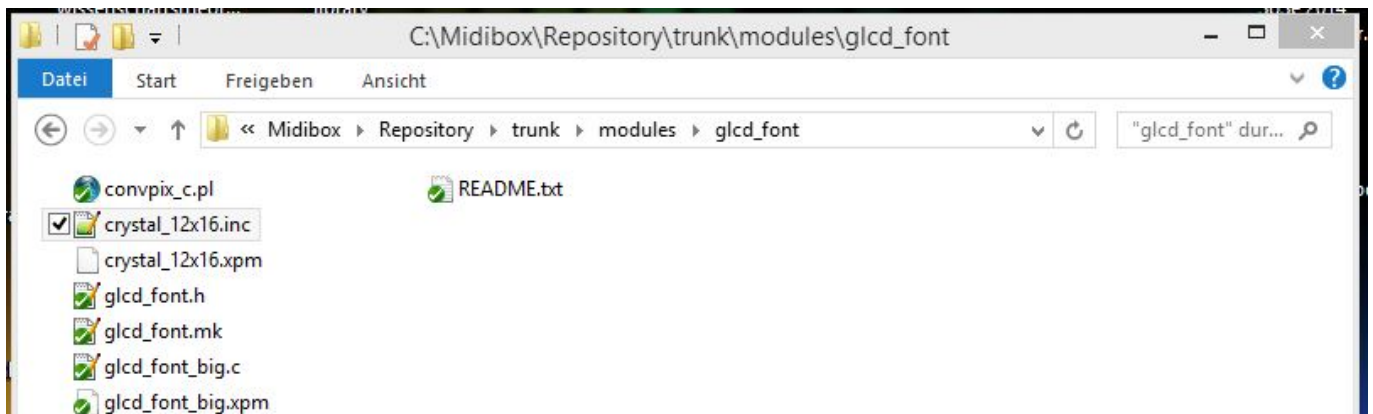
Next you have to open the image file (now "crystal_12x16.c") with the texteditor and fill in the necessary header information.
At line 3 include "mios32.h".
At line 5 give it a name (case sensitive!) and open a function.
At line 6 define the size of the icons and the offset (normally a width of 12 would need a char offset of 12. This means, MIOS is looking after 12 pixels for a new icon).

Don´t forgt the closing bracket and semicolon.

3. Implementing the font file into Midibox_NG

To implement the font file you have to modify several files in the repository.

First, inside ...\trunk\modules\glcd_font open the file "glcd_font.h" with the texteditor. Make an entry like in the screenshot to define an array pointer to the font.

Second, inside the same folder open "glcd_font.mk" with the texteditor. There the path to the font file has to be defined.
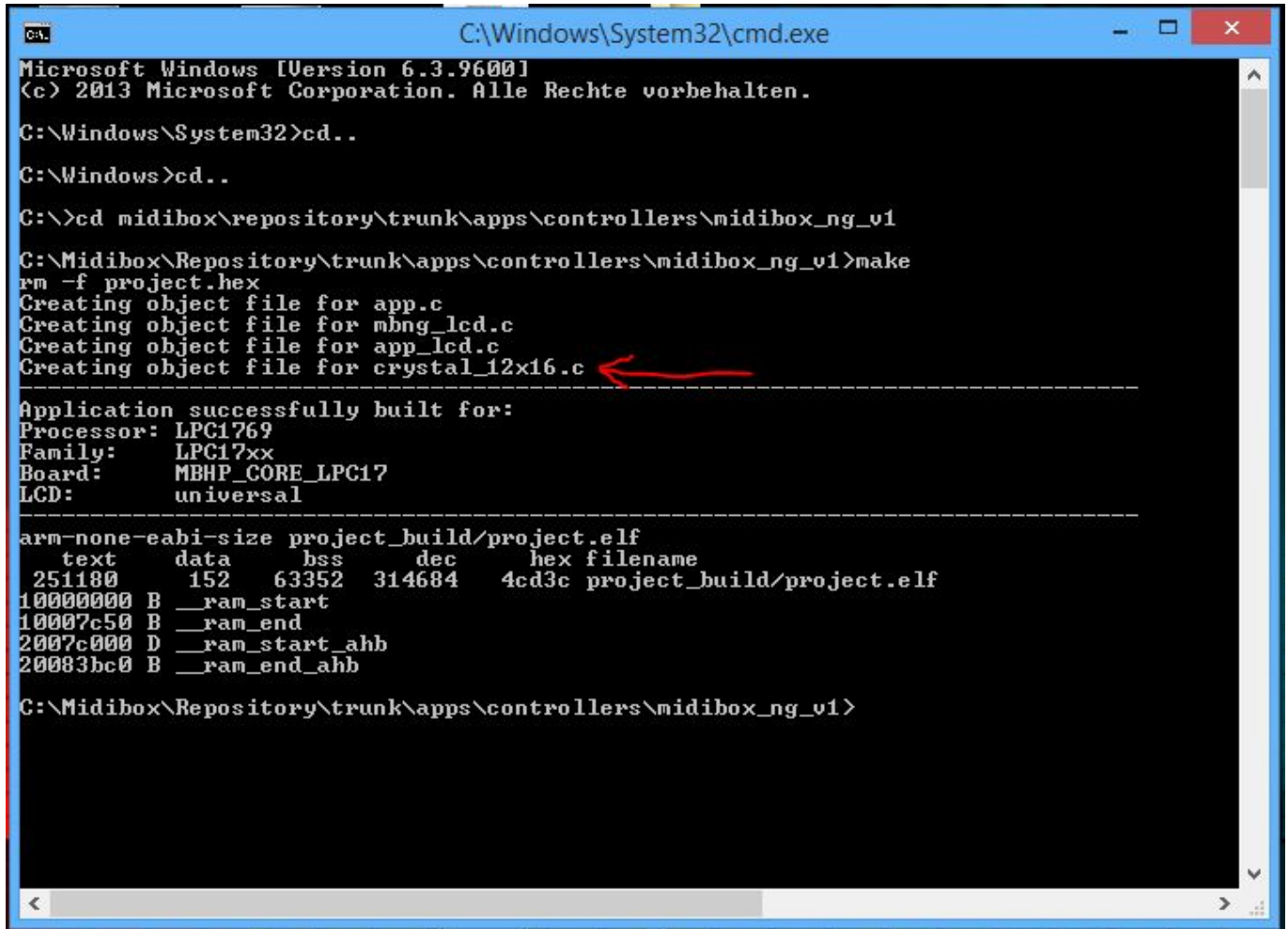
Third, a font selector has to be defined for Midibox_NG. Inside
…\trunk\apps\controllers\midibox_ng_v1\src open the file "mbng_lcd.c" with the texteditor. Scroll down a bit and find the function MBNG_LCD_Fontinit. Inside this function create a new switch for the font like in the screenshot. Any characters are available, but they can be used only once. Since some are already taken by the default fonts, just avoid them.
Also special characters which are already used by the firmware to display particular display elements (like "%" or "@") should be avoided too. Using those can cause your Midibox to crash.

The last step would be to recompile the firmware. If you have compiled the firmware on your system before and didn´t make any other changes than those presented in this tutorial, then the readout on the command window should look somewhat like this:

You can see that the files "app.c","mbng_lcd.c","app_lcd.c" have been updated and that your created fonts ("crystal_12x16.c" in my case) have been implemented.

Now the new fonts and icons can be used with your Midibox_NG. Congrats!

Examples

Here you can see the some custom fonts and icons I created in action.
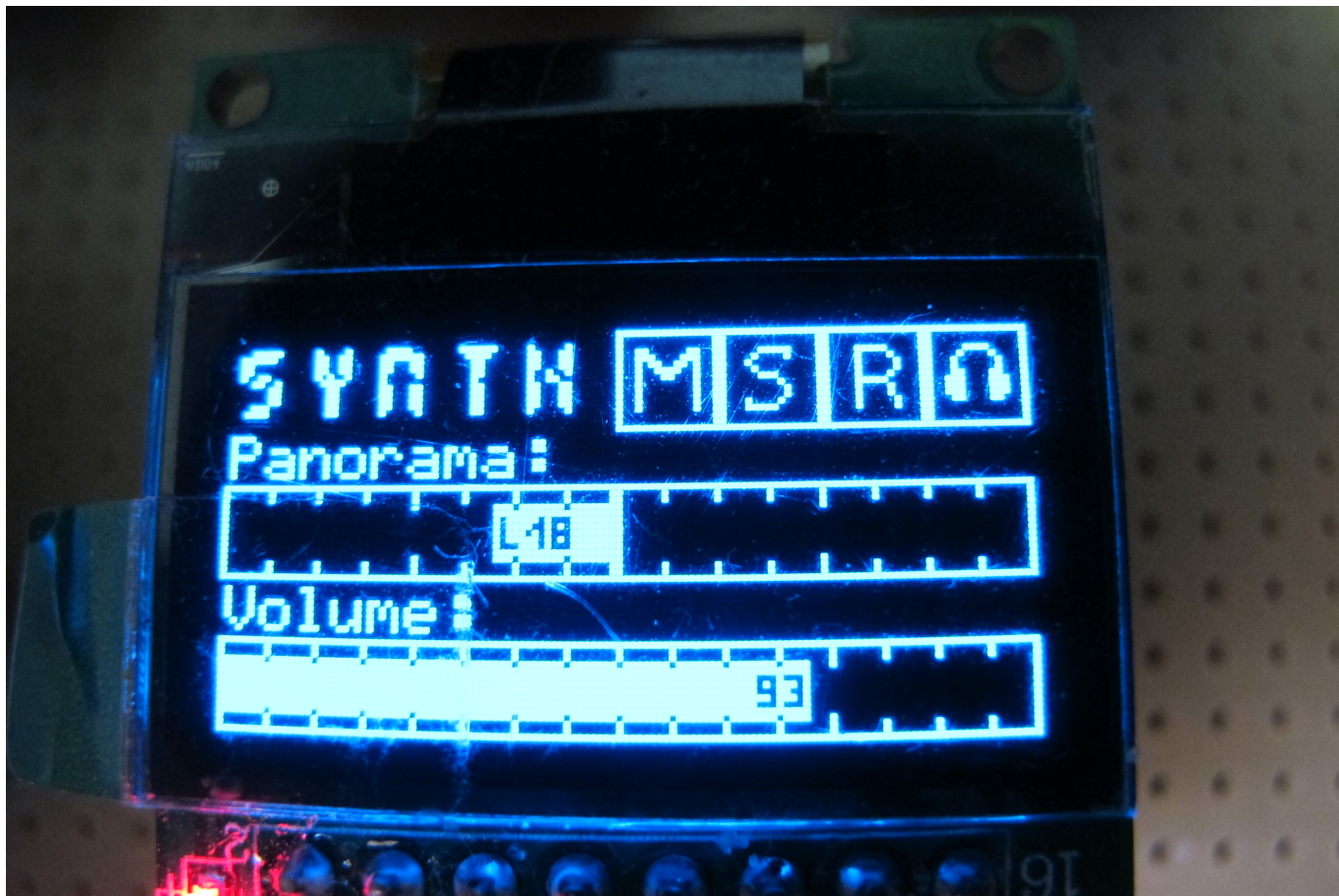
In this section I want to walk you through some possible usecases for your new fonts and show you some ways on how to get these integrated in your Midibox_NG.

Text

The track label you can see here is just a simple static text element but it demonstrates the font I implemented troughout this tutorial.

NGC:

```
LCD "&a@(1:1:1)synth"
```

The new font can be selected like any other font with a selector "&". In this case I defined "a" as the selector for this font.

Icons

The next thing I want to demonstrate is the use of custom icons like mute/solo/arm/monitor. This icons change depending on the status of the control element, so I used a NGL script.

NGL:

```
COND_LABEL mute
        COND= 0      "&mA"
        COND_ELSE    "&mB"
COND_LABEL solo
        COND= 0      "&mC"
        COND_ELSE    "&mD"
```

```
COND_LABEL rec
      COND= 0     "&mE"
      COND_ELSE   "&mF"
COND_LABEL monitor
      COND= 0     "&m6"
      COND_ELSE   "&m7"
```

NGC:

```
EVENT_BUTTON id=1  type=CC chn=1 cc=101 lcd_pos=1:6:1 label="^mute"
EVENT_BUTTON id=2  type=CC chn=1 cc=102 lcd_pos=1:7:1 label="^solo"
EVENT_BUTTON id=3  type=CC chn=1 cc=103 lcd_pos=1:8:1 label="^rec"
EVENT_BUTTON id=4  type=CC chn=1 cc=104 lcd_pos=1:9:1 label="^monitor"
```

In this case the icons are selected inside the NGL script and called from a NGC script by a specific name. The icons correspond to regular letters and digits (here "A,B,C,D,E,F,6,7") but the graphics for those are pulled from an icon font file (here selected by "&m"). The icons have a size of 16×16 pixels.
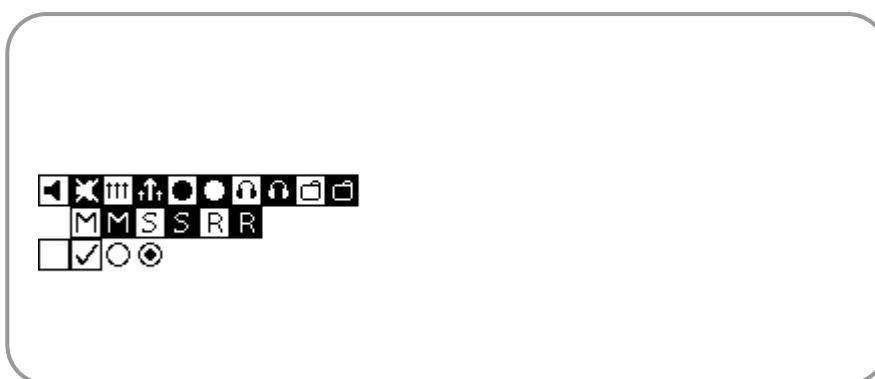
bars

Imo, one of the most useful usecases of custom fonts is to be able to display "high definition" bars on the display. In the example above the panorama and the volume bars have a resolution of 128 steps (corresponding to the assigned CC controllers). To realise this a NGL script has to be used again.

Also, since the fonts I created in this tutorial only have 96 usable characters, the HD bar has to be split up into two seperate fonts.

NGL:

```
COND_LABEL volume
      COND= 0     "&o0"
      COND= 1     "&o1"
      COND= 2     "&o2"
      ...
      COND= 65    "&p0"
      COND= 66    "&p1"
      COND= 67    "&p2"
      ...
      COND_ELSE   "FAIL"
```

NGC:

```
EVENT_ENC id=1 type=cc chn=1 cc=5 lcd_pos=1:1:4 label="^volume"
```

Inside the NGL script I created a conditional label for the controller that controls volume. For every possible value from 0-64 and 65-127 I told the Midibox to display another character and told it to use font "&o" for the first and font "&p" for the second half of the values.

The same principle aplies for the panorama bar.

Here are the corresponding fonts, the individual icons have a size of 128×16 pixels (WxH).
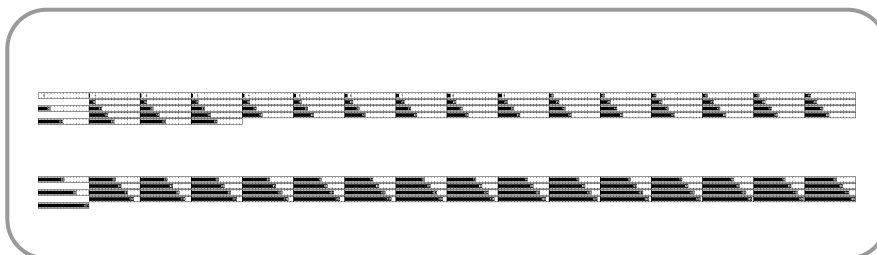
Known issues

1. I had some issues with icons that fill out all four corner pixels, they caused visible artifacts on other GLCDs. You can read about that here (forum link!!!). I can´t tell if that problem only existed on my hardware, if it was a software glitch or if this has been straighten out in the meantime by newer firmware versions. Unfortunatly I don´t have nearly enough time for my Midiboxes as I wish.

As a solution I avoided painting the four corner pixels and left them white.

(Yes, I know! My examples here have painted corner pixels. They´re from a time when I wasn´t fully aware of that issue.)

If you want to use only one GLCD, then this issue might not affect you at all.

Have Fun!

Ok, that´s it so far. If you have any questions, come across mistakes in this tutorial or have any other suggestions just contact me in the forum.

Disclaimer

I hope this page can be of help to other Midiboxers. I apologise for all typos, wrong technical terms and other linguistic misconducts on this page. It is still a second language to me…

My Regards

John E. Finster

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

Permanent link:
http://wiki.midibox.org/doku.php?id=john_e._finster&rev=1398684312

Last update: **2014/04/28 12:25**