How to create custom GLCD fonts/icons/bars 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, implementing custom fonts in the way it is presented in this tutorial is not and can´t be made part of the official midibox repository. So every time the firmware gets updated and you want to keep your custom fonts, all the changes in step 3 of this tutorial has to be repeated.
I can´t tell for sure, but I believe, if you update the repository the files changed by you don´t get updated. To assure that everything works after an update I recommend deleting all the changed files before the update and then getting all the new files again.

Forth, this tutorial only aplies to graphical lcds.

Ok, let´s get to down to business

Disclaimer

Creating custom fonts/icons/bars is not part of the official repository. Any use of this knowledge presented in this tutorial will be happening at your own risk. If your Midibox suddenly blows up or melts down, it won´t be my fault.

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.

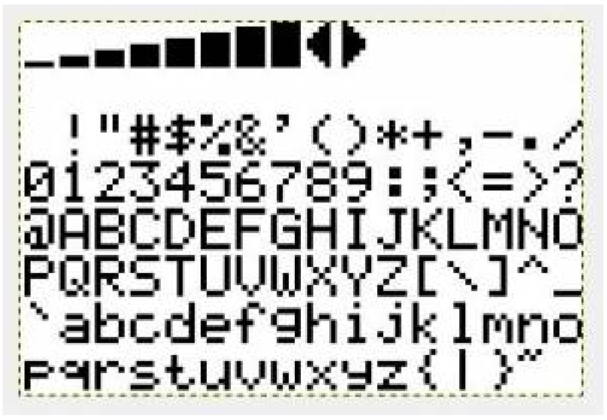
- All the tools necessary to compile the firmware (as stated on the ucapps programming site) including the complete Midibox repository.

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

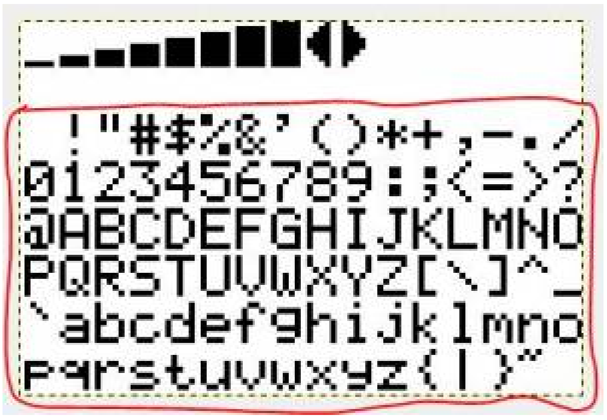
When you have not done so already, the Midibox_NG firmware should be compiled at least once before you start to mess with the code. On the one hand, you have asured that your system can compile the firmware without problems. That makes looking for errors much easier later . On the other hand, the Windows 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 is a font image that looks like this:

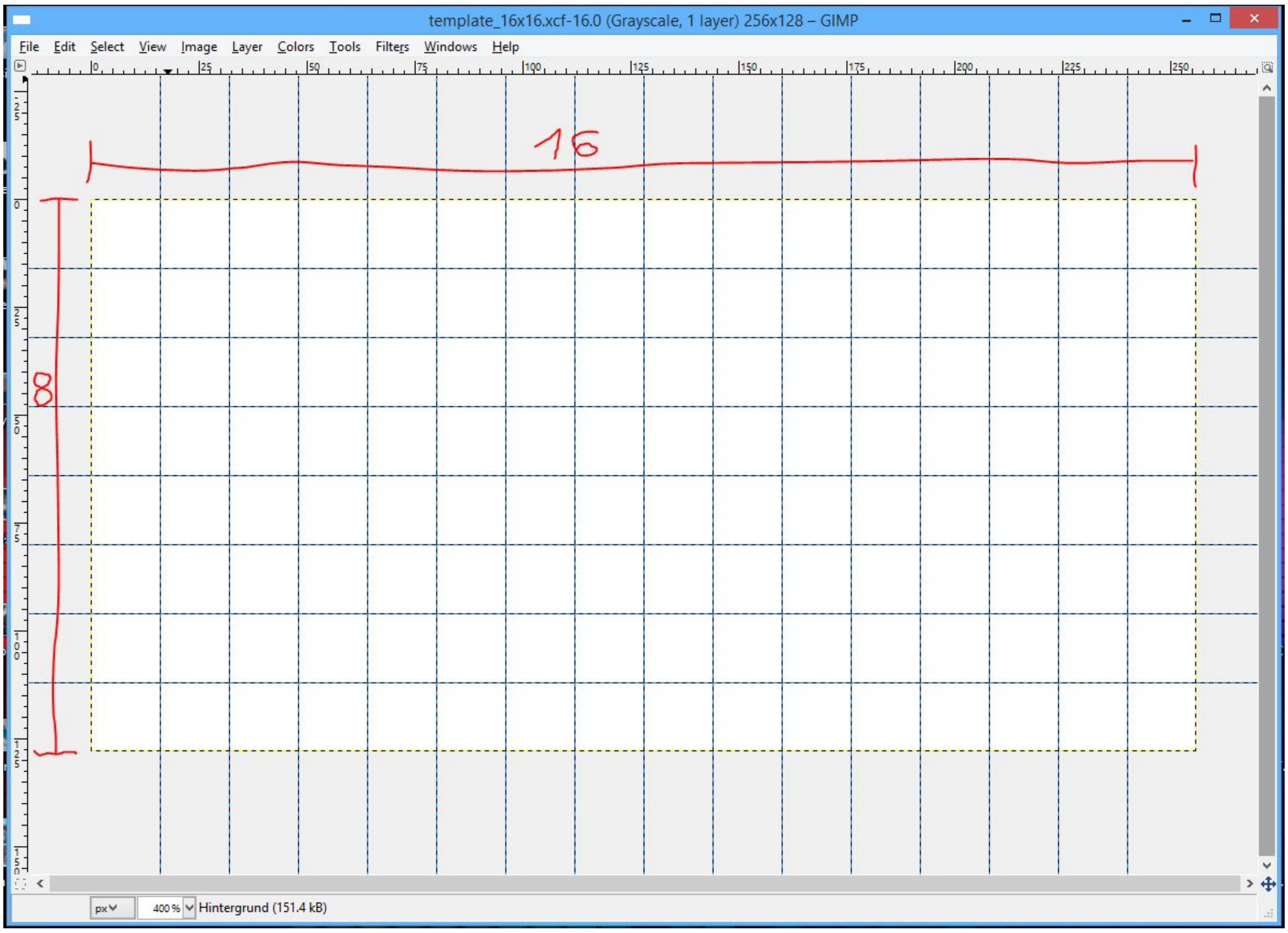


This is an original font file from TK. It consists of 8 rows and 16 columns, that makes 128 cells, so technically 128 icons can be put in. The top row is used for the default bar and some special icons used by the Midibox. The second row is empty for it´s not used by the Midibox_NG firmware (afaik). You could also manipulate the default bar and arrow icons, but I will focus on creating usable icons in the six lower rows.

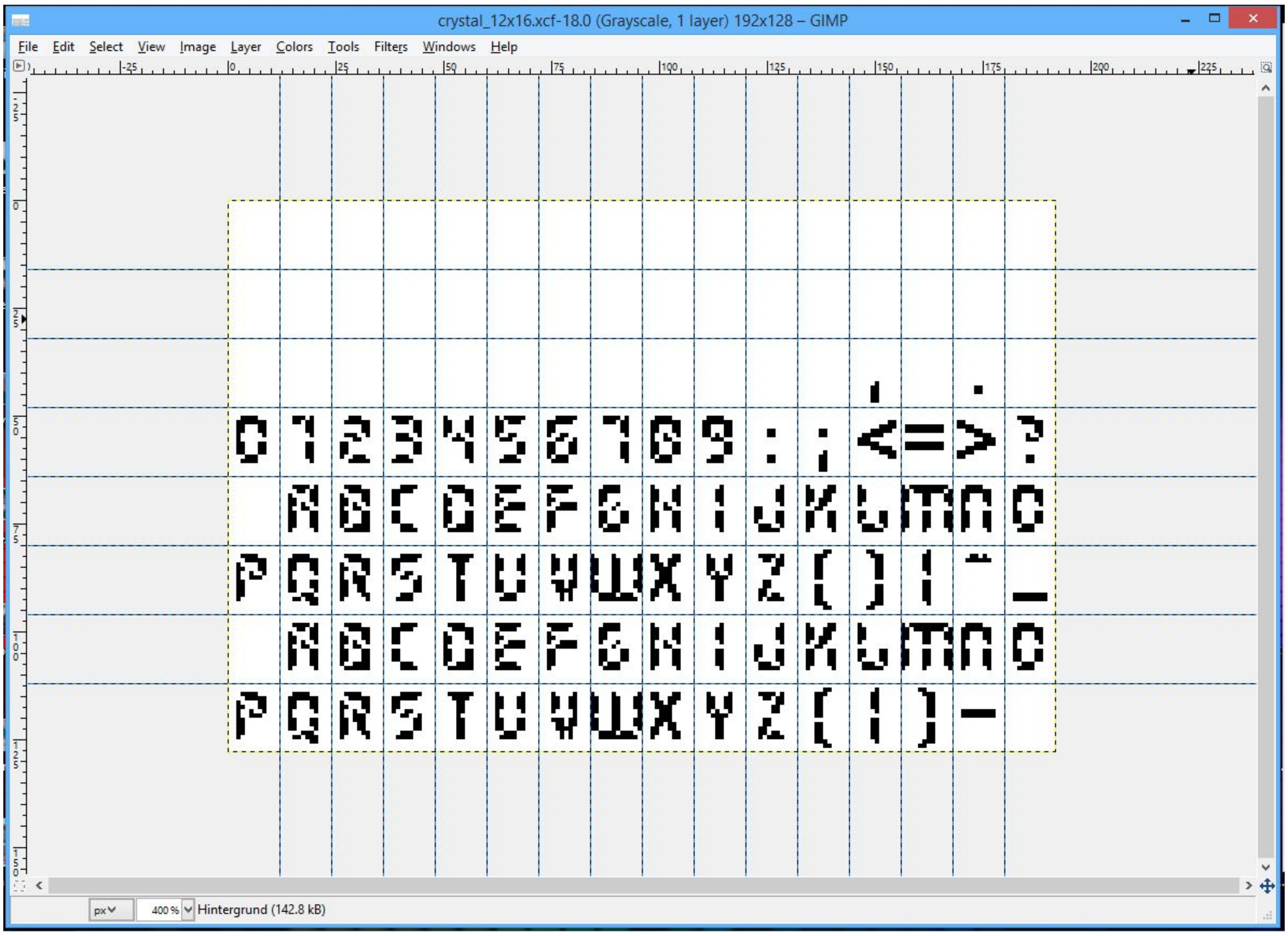


So, at first you need an empty image with overlaying guidelines, creating 8×16 evenly sized cells. Every cell has to have a height of a multiply of 8 pixels (8, 16, 24, …).
E.g. if you want to have icons with a size of 16×16 pixels, your image would be 256×128 pixels (WxH)

big.
If you want to create high resolution bars (in a later example, see below) with the size of 128×16 pixels (WxH), then your empty image has to be 2048×128 pixels (WxH) big.


template_16x16.xcf-16.0 (Grayscale, 1 layer) 256x128 – GIMP
File Edit Select View Image Layer Colors Tools Filters Windows Help
16
8
px
400 %
Hintergrund (151.4 kB)

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.

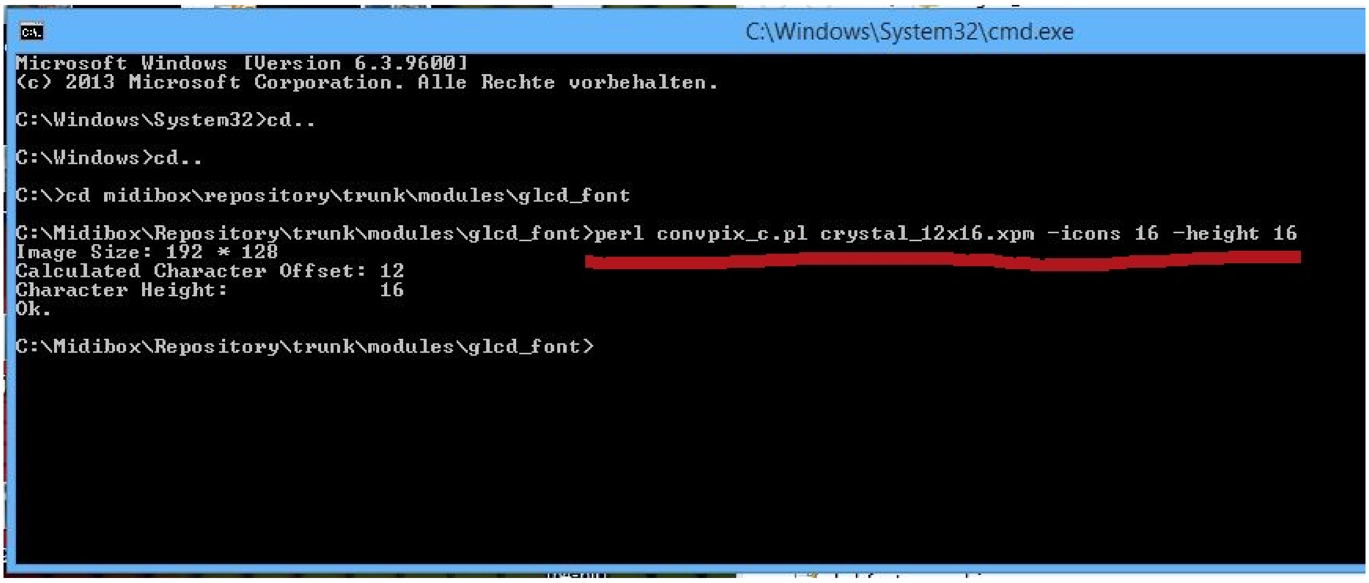

crystal_12x16.xcf-18.0 (Grayscale, 1 layer) 192x128 – GIMP
File Edit Select View Image Layer Colors Tools Filters Windows Help
px
400 %
Hintergrund (142.8 kB)

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

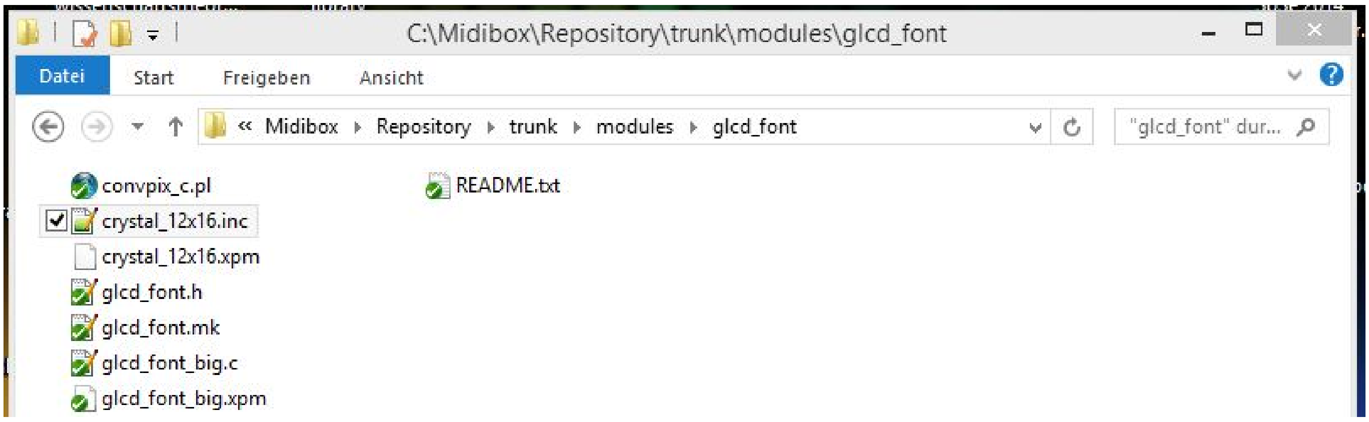
For the next step you have to move the image *.xpm file into the MIOS32 repository (*…\trunk\modules \glcd_font*). Inside this folder there are the other Midibox fonts. Also there is a perl script called "convpix_c.pl". You have to use it to convert the *xpm file to a MIOS32 compatible font file. To 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.

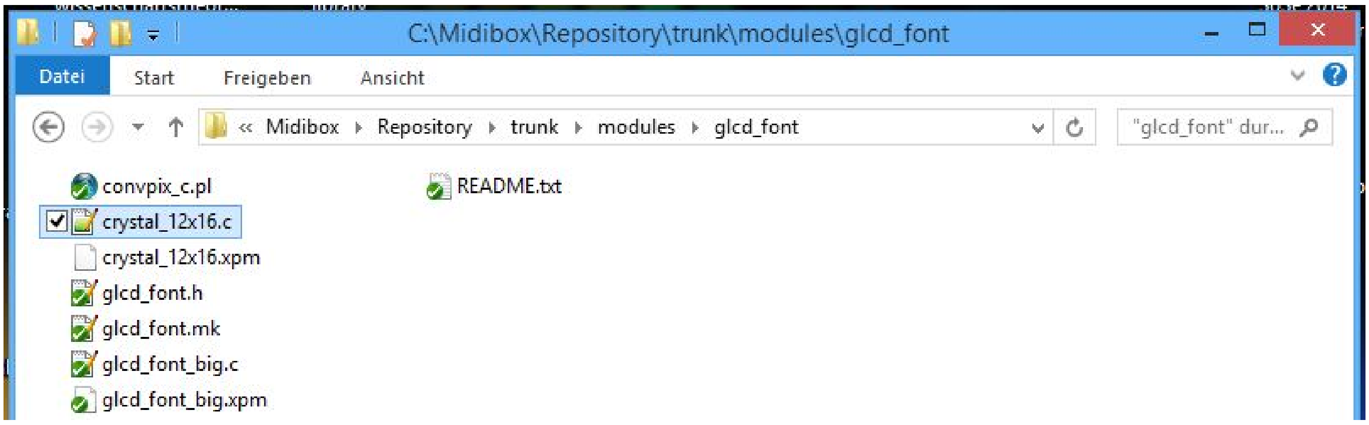


```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. Alle Rechte vorbehalten.

C:\Windows\System32>cd..

C:\Windows>cd..

C:\>cd midibox\repository\trunk\modules\glcd_font

C:\Midibox\Repository\trunk\modules\glcd_font>perl convpix_c.pl crystal_12x16.xpm -icons 16 -height 16
Image Size: 192 * 128
Calculated Character Offset: 12
Character Height:            16
Ok.

C:\Midibox\Repository\trunk\modules\glcd_font>
```

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.

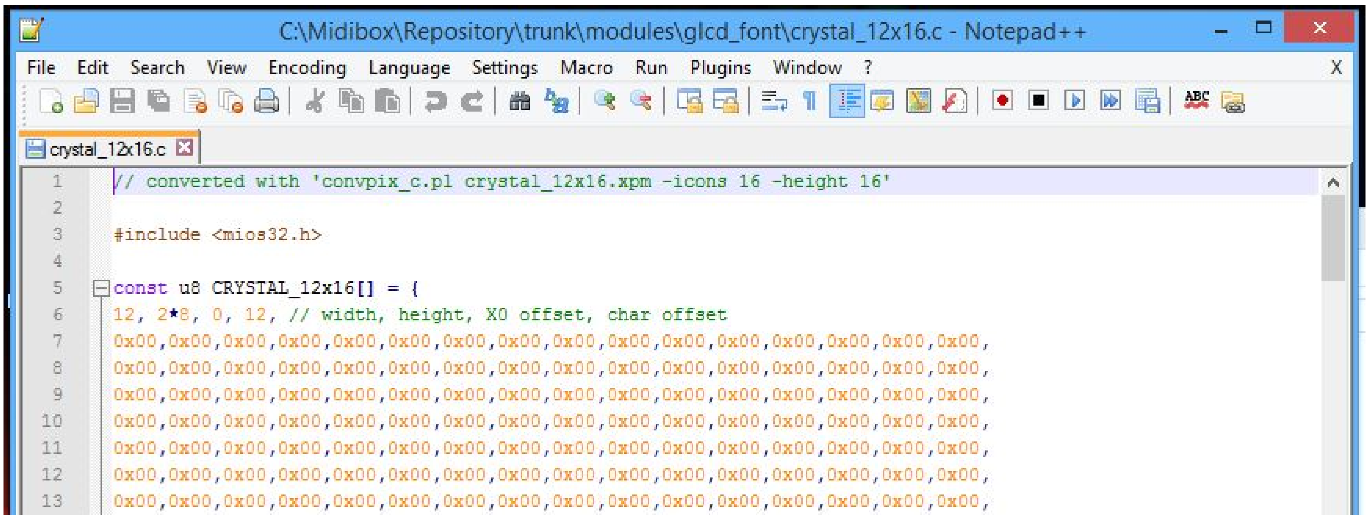
What we now have is an *.inc file of the image which we´re going to change to a *.c file simply by renaming it. Windows will complain eventually, but we can ignore that.





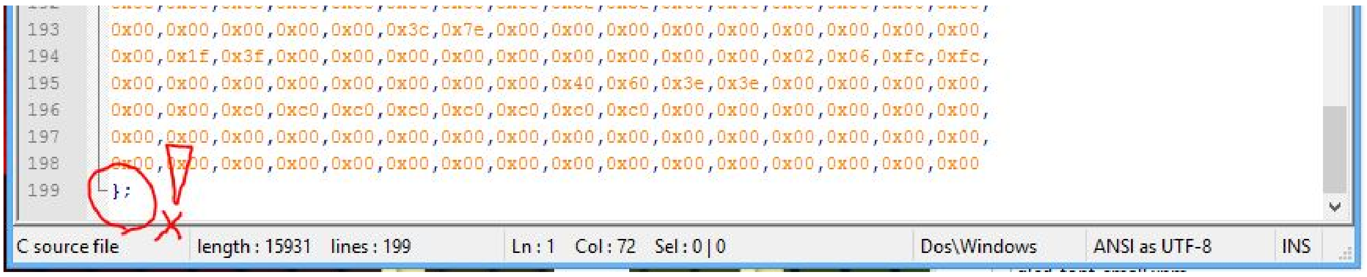
Next you have to open the image file (now “crystal_12x16.c”) with the texteditor and fill in the necessary header information.
At line 3 include “mios32.h”.
At line 5, after “const u8 ” give it a name (case sensitive!) and open a function.
At line 6 define the size of the icons and the offset (normally a width of 12 would need a char offset of 12. This means, MIOS is looking after 12 pixels for a new icon). Define the height of the font as a

multiply of 8.


C:\Midibox\Repository\trunk\modules\glcd_font\crystal_12x16.c - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
crystal_12x16.c

```
// converted with 'convpix_c.pl crystal_12x16.xpm -icons 16 -height 16'

#include <mios32.h>

const u8 CRYSTAL_12x16[] = {
12, 2*8, 0, 12, // width, height, X0 offset, char offset
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
```

Don´t forgt the closing bracket and semicolon.

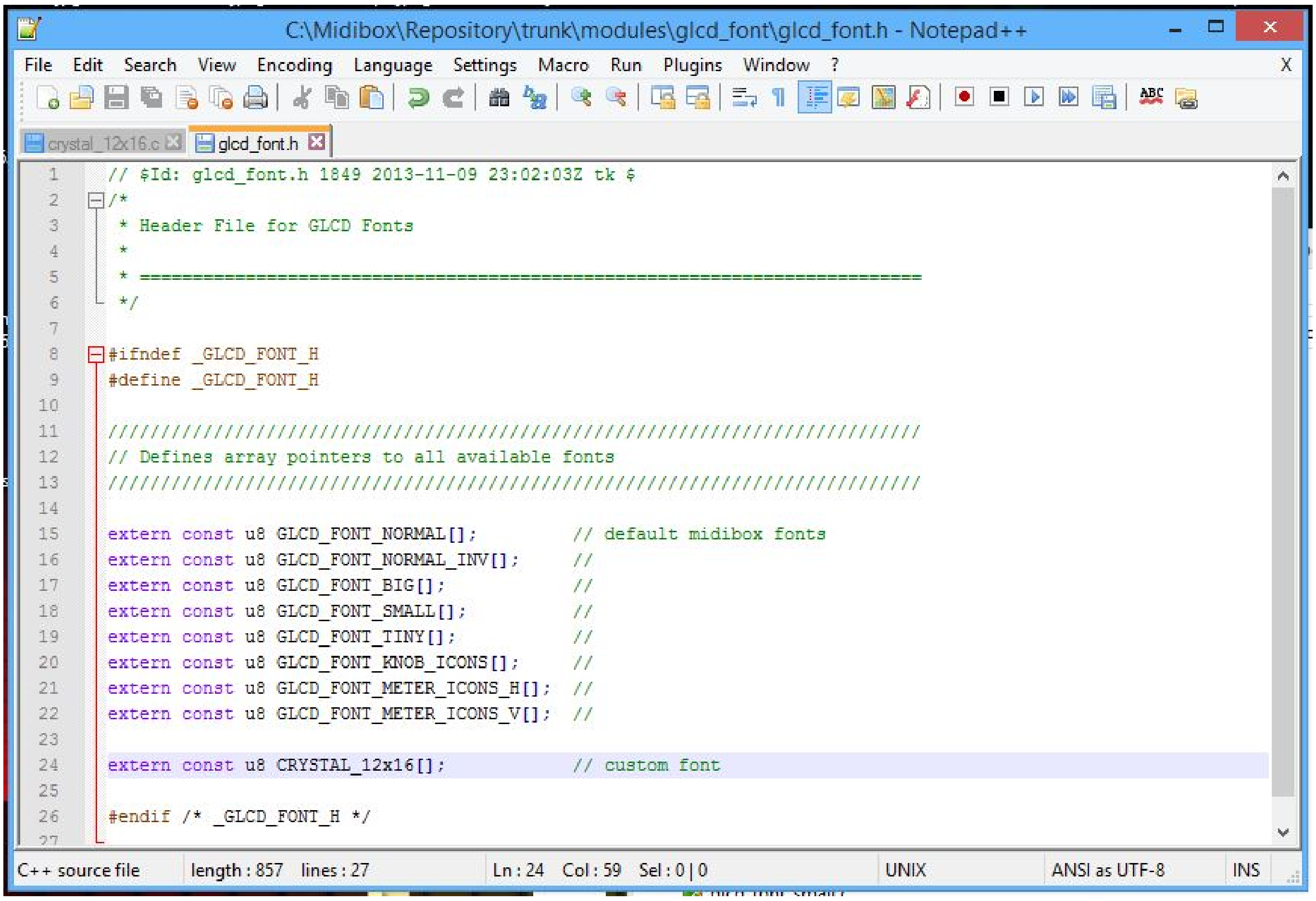


```
0x00,0x00,0x00,0x00,0x00,0x3c,0x7e,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x1f,0x3f,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x02,0x06,0xfc,0xfc,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x40,0x60,0x3e,0x3e,0x00,0x00,0x00,0x00,
0x00,0x00,0xc0,0xc0,0xc0,0xc0,0xc0,0xc0,0xc0,0xc0,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
};
```

3. Implementing the font file into Midibox_NG

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

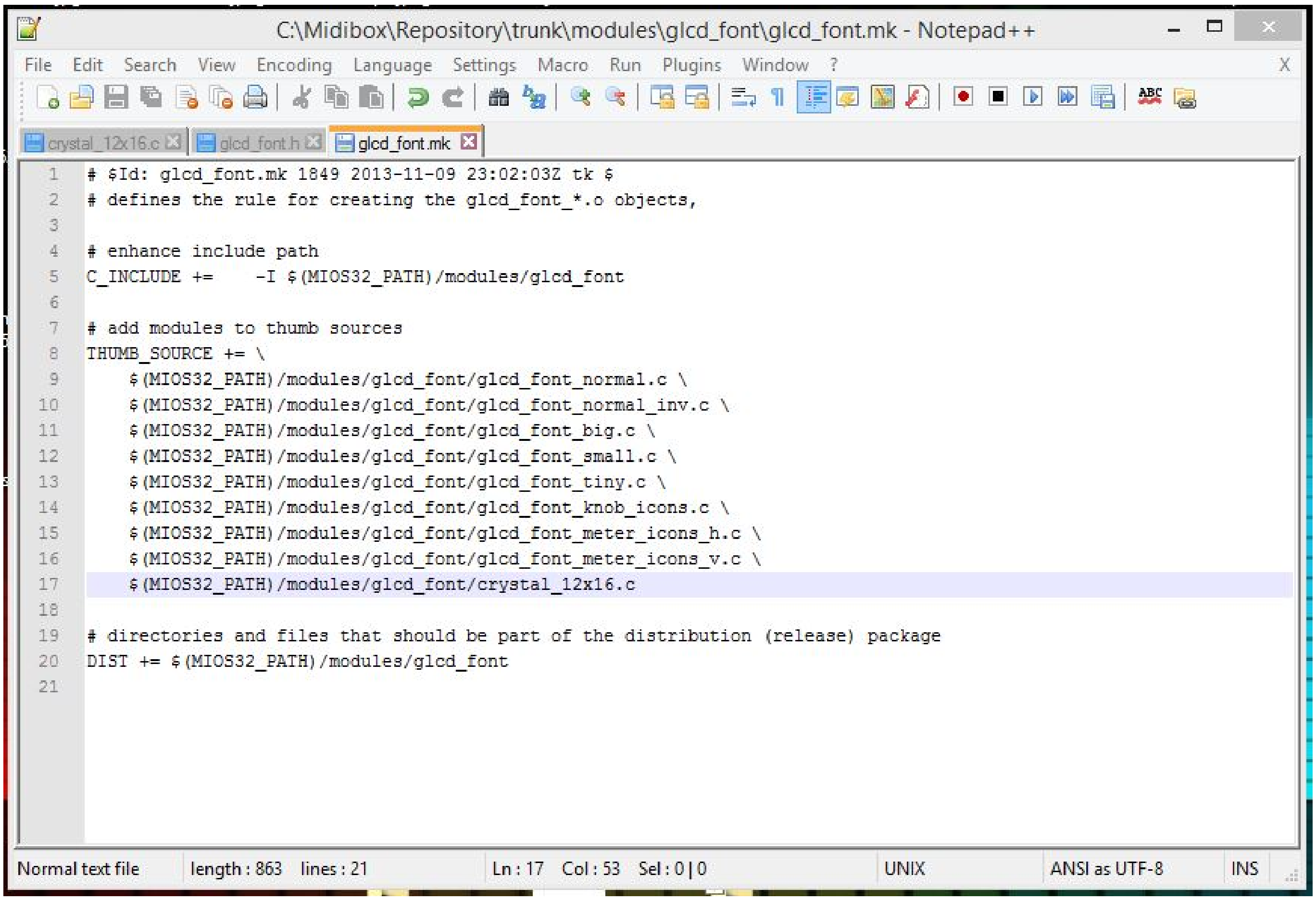
First, inside …*trunk**modules**glcd_font* open the file "glcd_font.h" with the texteditor. Make an entry like in the screenshot to define an array pointer to the font.


C:\Midibox\Repository\trunk\modules\glcd_font\glcd_font.h - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
crystal_12x16.c
glcd_font.h

```
// $Id: glcd_font.h 1849 2013-11-09 23:02:03Z tk $
/*
 * Header File for GLCD Fonts
 *
 * ==========================================================================
 */

#ifndef _GLCD_FONT_H
#define _GLCD_FONT_H

/////////////////////////////////////////////////////////////////////////////
// Defines array pointers to all available fonts
/////////////////////////////////////////////////////////////////////////////

extern const u8 GLCD_FONT_NORMAL[];         // default midibox fonts
extern const u8 GLCD_FONT_NORMAL_INV[];     //
extern const u8 GLCD_FONT_BIG[];            //
extern const u8 GLCD_FONT_SMALL[];          //
extern const u8 GLCD_FONT_TINY[];           //
extern const u8 GLCD_FONT_KNOB_ICONS[];     //
extern const u8 GLCD_FONT_METER_ICONS_H[];  //
extern const u8 GLCD_FONT_METER_ICONS_V[];  //

extern const u8 CRYSTAL_12x16[];            // custom font

#endif /* _GLCD_FONT_H */
```

C++ source file length : 857 lines : 27 Ln : 24 Col : 59 Sel : 0 | 0 UNIX ANSI as UTF-8 INS

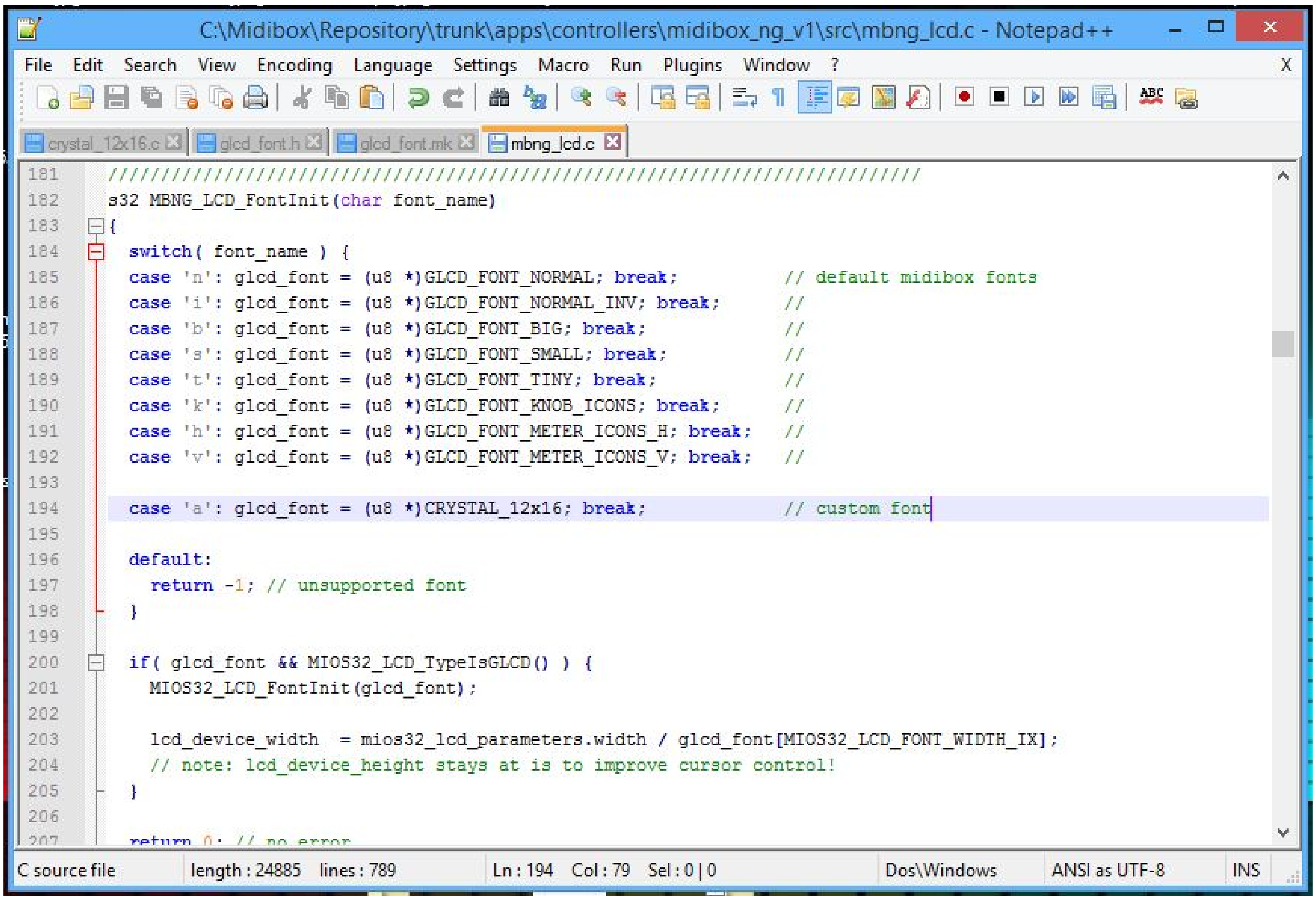
Second, inside the same folder open “glcd_font.mk” with the texteditor. There the path to the font file has to be defined.


C:\Midibox\Repository\trunk\modules\glcd_font\glcd_font.mk - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
crystal_12x16.c
glcd_font.h
glcd_font.mk

```
# $Id: glcd_font.mk 1849 2013-11-09 23:02:03Z tk $
# defines the rule for creating the glcd_font_*.o objects,

# enhance include path
C_INCLUDE +=    -I $(MIOS32_PATH)/modules/glcd_font

# add modules to thumb sources
THUMB_SOURCE += \
    $(MIOS32_PATH)/modules/glcd_font/glcd_font_normal.c \
    $(MIOS32_PATH)/modules/glcd_font/glcd_font_normal_inv.c \
    $(MIOS32_PATH)/modules/glcd_font/glcd_font_big.c \
    $(MIOS32_PATH)/modules/glcd_font/glcd_font_small.c \
    $(MIOS32_PATH)/modules/glcd_font/glcd_font_tiny.c \
    $(MIOS32_PATH)/modules/glcd_font/glcd_font_knob_icons.c \
    $(MIOS32_PATH)/modules/glcd_font/glcd_font_meter_icons_h.c \
    $(MIOS32_PATH)/modules/glcd_font/glcd_font_meter_icons_v.c \
    $(MIOS32_PATH)/modules/glcd_font/crystal_12x16.c

# directories and files that should be part of the distribution (release) package
DIST += $(MIOS32_PATH)/modules/glcd_font
```

Normal text file length : 863 lines : 21 Ln : 17 Col : 53 Sel : 0 | 0 UNIX ANSI as UTF-8 INS

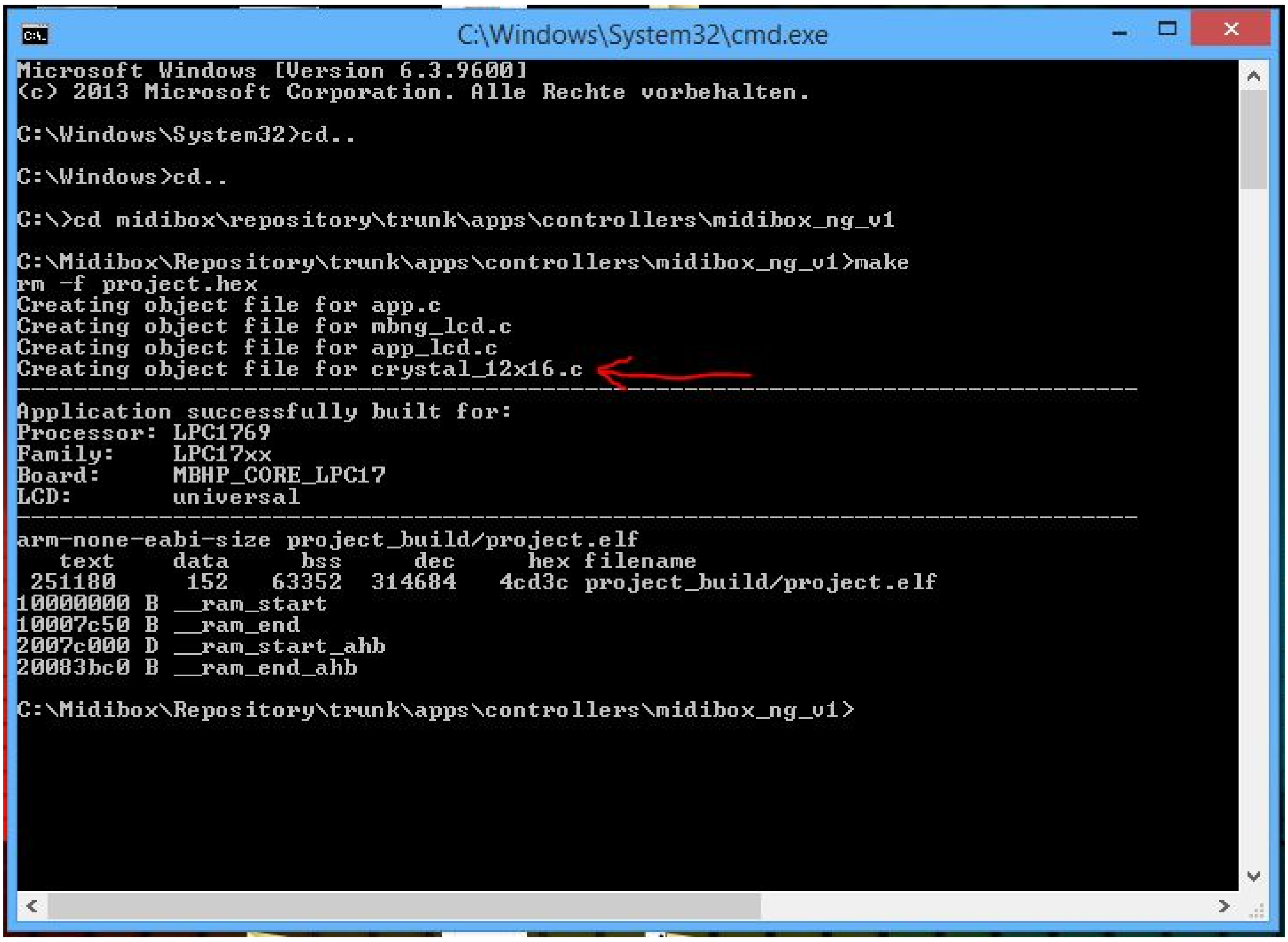
Third, a font selector has to be defined for Midibox_NG. Inside *…\trunk\apps\controllers\midibox_ng_v1\src* open the file "mbng_lcd.c" with the texteditor. Scroll down a bit and find the function MBNG_LCD_Fontinit. Inside this function create a new switch for the font like in the screenshot. Any characters 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.


C:\Midibox\Repository\trunk\apps\controllers\midibox_ng_v1\src\mbng_lcd.c - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
crystal_12x16.c
glcd_font.h
glcd_font.mk
mbng_lcd.c

```
/////////////////////////////////////////////////////////////////////////////
s32 MBNG_LCD_FontInit(char font_name)
{
  switch( font_name ) {
  case 'n': glcd_font = (u8 *)GLCD_FONT_NORMAL; break;          // default midibox fonts
  case 'i': glcd_font = (u8 *)GLCD_FONT_NORMAL_INV; break;      //
  case 'b': glcd_font = (u8 *)GLCD_FONT_BIG; break;             //
  case 's': glcd_font = (u8 *)GLCD_FONT_SMALL; break;           //
  case 't': glcd_font = (u8 *)GLCD_FONT_TINY; break;            //
  case 'k': glcd_font = (u8 *)GLCD_FONT_KNOB_ICONS; break;      //
  case 'h': glcd_font = (u8 *)GLCD_FONT_METER_ICONS_H; break;   //
  case 'v': glcd_font = (u8 *)GLCD_FONT_METER_ICONS_V; break;   //

  case 'a': glcd_font = (u8 *)CRYSTAL_12x16; break;             // custom font

  default:
    return -1; // unsupported font
  }

  if( glcd_font && MIOS32_LCD_TypeIsGLCD() ) {
    MIOS32_LCD_FontInit(glcd_font);

    lcd_device_width  = mios32_lcd_parameters.width / glcd_font[MIOS32_LCD_FONT_WIDTH_IX];
    // note: lcd_device_height stays at is to improve cursor control!
  }

  return 0; // no error
```

```
C source file   length : 24885   lines : 789   Ln : 194   Col : 79   Sel : 0 | 0   Dos\Windows   ANSI as UTF-8   INS
```

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:



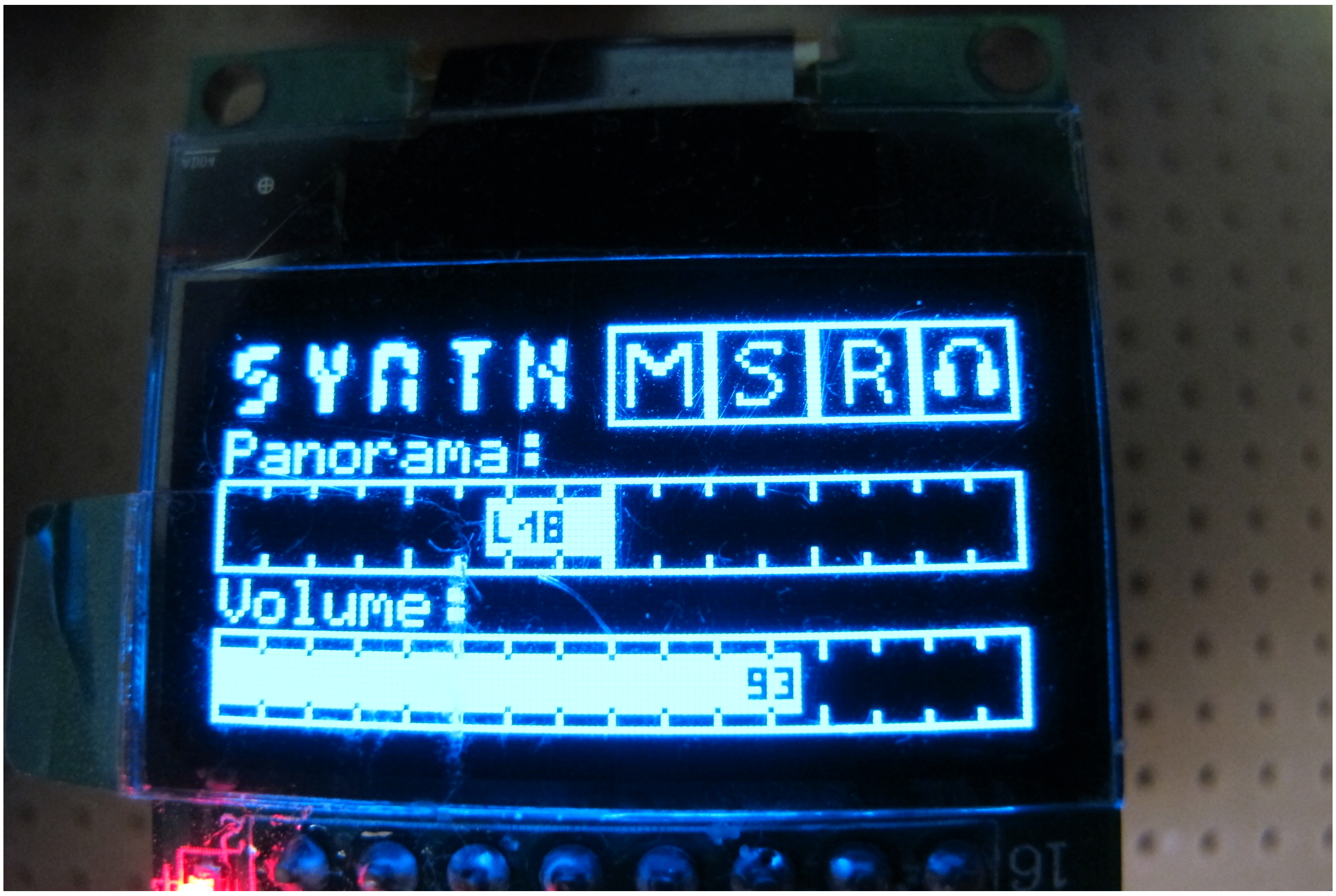
```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. Alle Rechte vorbehalten.

C:\Windows\System32>cd..

C:\Windows>cd..

C:\>cd midibox\repository\trunk\apps\controllers\midibox_ng_v1

C:\Midibox\Repository\trunk\apps\controllers\midibox_ng_v1>make
rm -f project.hex
Creating object file for app.c
Creating object file for mbng_lcd.c
Creating object file for app_lcd.c
Creating object file for crystal_12x16.c
-------------------------------------------------------------------------------
Application successfully built for:
Processor: LPC1769
Family:    LPC17xx
Board:     MBHP_CORE_LPC17
LCD:       universal
-------------------------------------------------------------------------------
arm-none-eabi-size project_build/project.elf
   text    data     bss     dec     hex filename
 251180     152   63352  314684   4cd3c project_build/project.elf
10000000 B __ram_start
10007c50 B __ram_end
2007c000 D __ram_start_ahb
20083bc0 B __ram_end_ahb

C:\Midibox\Repository\trunk\apps\controllers\midibox_ng_v1>
```

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:

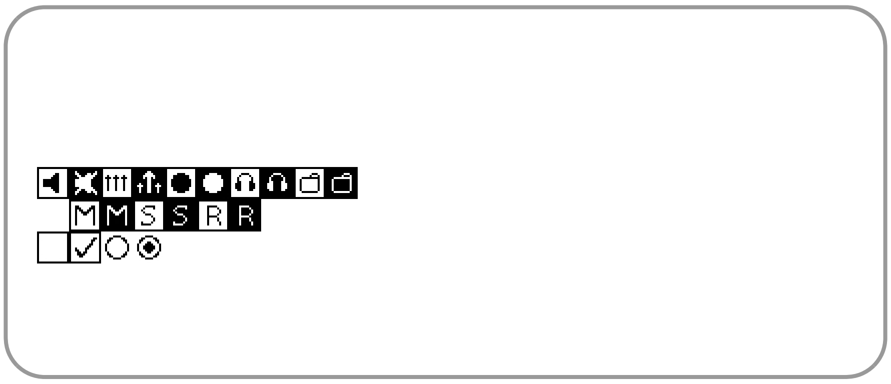
```
COND_LABEL mute
      COND =0     "&mA"
      COND_ELSE   "&mB"
COND_LABEL solo
```

```
      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"
```

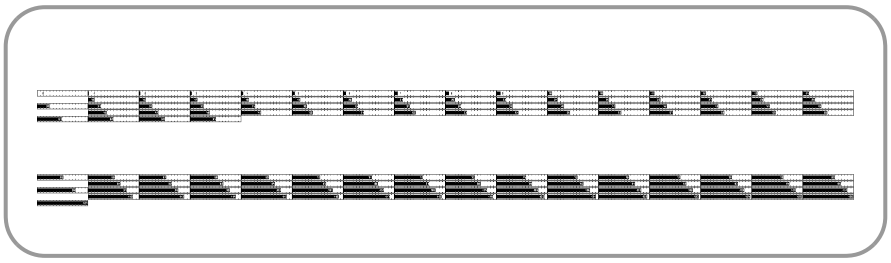
NGC:

```
       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.

2. Some special characters, which are already used by the midibox firmware to display certain elements (like “%” or “@”) can cause the midibox to crash if they´re placed next to a font selector (like “&o@”) or if they themselves are used as font selectors (like “&@”). I can´t totally explain why this is the case.
I recommend avoiding the slots in the font file where those special characters normally would sit and also using them as a font selector

Custom fonts database

Here some fonts made by other can be downloaded.

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.

Happy designing!
Greets
John E. Finster
