GPIO on the Olimex A20-OlinuXino Micro with Linux

(Kernel version 3.3.0+ and 3.4.67+)

A guide with Shell and C examples

Dr. Guido Pelz
DM1GP / PY6ZGP

www.dm1gp.com and www.py6zgp.com

26.9.2014

Table of Contents

1. Motivation, Scope and Disclaimer

I'm using GPIO on the A20 for over a year now. With the new image with kernel 3.4.67+ the numbering and the names of the GPIO devices has been changed. This caused me a lot of trouble, because I needed to change my software as well as my configuration.

The documentation on the A20 GPIO under Linux is still the old one and there is no hint how to do it correctly. As the documentation is outdated, I did some research myself. The result is this document, covering a little more than just GPIO.

What is this document about and what can you expect?

- Explains how to use GPIO (under both kernel versions)
- Explains how to change the port configuration
- Examples in Shell and C
- All just for Linux

You are wrong, If you expect

- code ready to use in your application
- Python code
- Android stuff

So, I'll not answer questions about Android and/or Python. Not because I don't want, just because I have no idea about both subjects!

I tried my best, but I'm absolutely sure, that there are some errors within this document and/or the code given here.

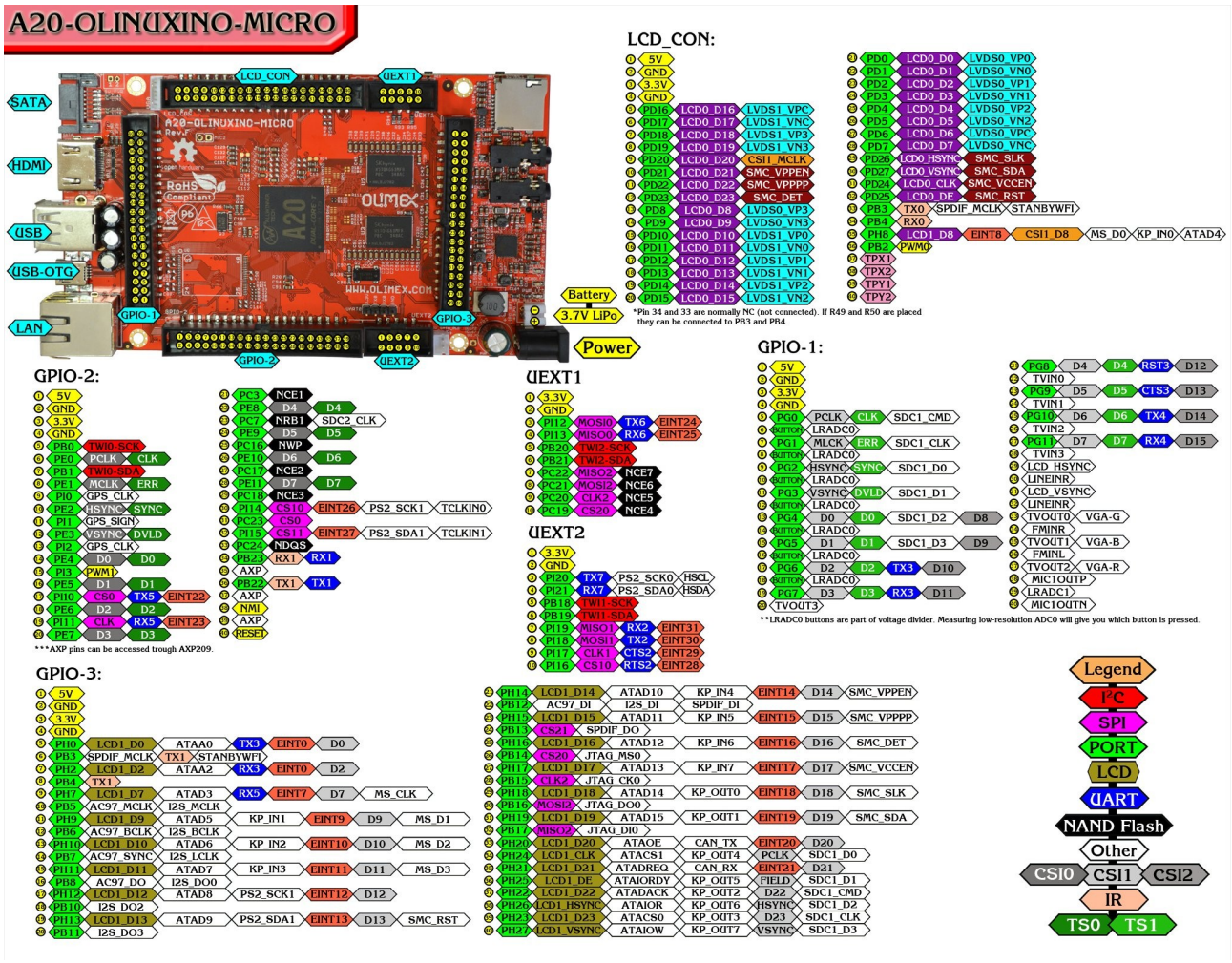
Therefore, I'll not take any responsibility for anything going wrong, any damage on your system or anything else related to this document. This document come as it is!

But you are welcome to improve this document. So please send your comments, improvements, or corrections.

Have fun and I hope, that this is helpful to somebody!

2. Processor Ports and Connectors

This is the official image of all the A20 I/O connectors. If you are just familiar with the kernel 3.3.0+ style of accessing GPIO, please have a closer look at the naming scheme of the pins.



Official image from https://www.olimex.com/wiki/A20-OlinuXino-MICRO.

As you can see, the green spots are named like

Pxnn	with x is a letter A, B, C, D, E, F, G, H, or I, and nn a number between 0 and 27 (depending on the chip)

The letter refers to the GPIO chip involved and the number to the actual port. These names are under kernel 3.4.67+ part of the device name! The number of ports per chip are as follows:

PA:	18 ports	PB:	24 ports	PC:	25 ports
PD:	28 ports	PE:	12 ports	PF:	6 ports
PG:	12 ports	PH:	28 ports	PI:	22 ports

All together 175 ports. But not all of them you can use for I/O. The majority is used for the system itself. 28 ports are reserved for connecting a LCD. Below, I will show you how to modify the configuration to use these port for your purposes as well (without LCD of cause).

With kernel 3.4.67+ the configuration only allows to access 75 GPIO ports and the ports on the LCD connector are not included.

2.1 GPIO numbering under kernel 3.3.0+

The most complete source of information (up to my knowledge) about the GPIO port numbering under kernel 3.3.0+ is this forum topic:

https://www.olimex.com/forum/index.php?topic=1514.0

The user Ruler uploaded the following image showing the discovered ports:

LCD

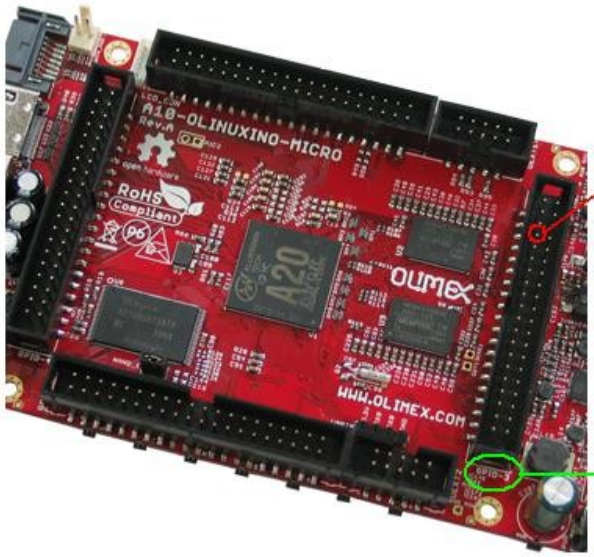
GPIO	GND	GND	102	104	106	108	94	96	98	100	86	88	90	92				26		
Pin	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40
Pin	1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	33	35	37	39
GPIO	5v	3,3v	101	103	105	107	93	95	97	99	85	87	89	91						

GND
5V
3,3V

GPIO	Pin	Pin	GPIO
	40	39	
	38	37	
	36	35	
	34	33	
	32	31	
	30	29	
	28	27	160
	26	25	159
	24	23	158
	22	21	157
	20	19	156
	18	17	155
	16	15	154
	14	13	153
	12	11	152
	10	9	151
	8	7	150
	6	5	149
GND	4	3	3,3v
GND	2	1	5v

GPIO-4

166 best guess
165 best guess
164 best guess
163 best guess
162 best guess
161 best guess



GPIO	Pin	Pin	GPIO
5v	1	2	GND
3,3v	3	4	GND
167	5	6	27
169	7	8	28
174	9	10	29
176	11	12	30
177	13	14	31
	15	16	32
179	17	18	34
180	19	20	35
181	21	22	36
182	23	24	37
183	25	26	38
184	27	28	39
185	29	30	40
186	31	32	41
187	33	34	191
	35	36	192
189	37	38	193
190	39	40	194

Led1

GPIO-3

best guess
25 24

GPIO											212	211	204	203	202	201			3,3v	5v
Pin	39	37	35	33	31	29	27	25	23	21	19	17	15	13	11	9	7	5	3	1
Pin	40	38	36	34	32	30	28	26	24	22	20	18	16	14	12	10	8	6	4	2
GPIO					216	215													GND	GND

GPIO-2

Be careful! The names of the GPIO connectors are not the same as on the board itself.

3. The Kernels 3.3.0+ and 3.4.67+ images

If you have worked with GPIO under kernel 3.3.0+, the first observation after trying to use GPIO under kernel 3.4.67+ will be, that the names of the devices has changed:

	Kernel 3.3.0+	Kernel 3.4.67+
	`/sys/class/gpio/gpioXXX`	`/sys/class/gpio/gpioYYY_pCNN`
Example:	`/sys/class/gpio/gpio12`	`/sys/class/gpio/gpio47_ph2`

with

XXX	a number between 0 and 200+
YYY	a number between 1 and 75 on delivery (we will see how to have more)
C	a letter: a, b, c, d, e, f, h, h, or i (lower case, depending on the GPIO chip involved)
NN	a number between 0 and 27 (depending on the GPIO chip – see chapter 2)

Even the numbers of the GPIO ports are not the same between the two kernel versions!

If you have not taken the device names directly from the directory after exporting them, you software running smoothly under kernel 3.3.0+ will definitely not run under kernel 3.4.67+ – just because of the different device names and the completely different numbering.

So, if you like to use kernel 3.4.67+, you might need to update your software but definitely you need to update your configuration.

3.1 Advantages, and Disadvantages of each kernel

This is simply my point of view! You might have a completely different opinion.

Kernel 3.3.0+:
+ device names only uses numbers – no need to take care of the chip
  This makes it easy to predict the device name.
+ by default you can access all GPIO ports without changing the configuration
  of your system
- you could easily use a port used by systems sub-system
- the port numbering can not be changed

Kernel 3.4.67+:
+ the device name is transparent to the GPIO chip
+ you can only access those ports given in the system's configuration
+ the port numbering is more convenient (port numbers are ina row for each connector)
+ the numbering of the GPIO ports can be changed easily
- After export, you need to read the device name or store all devices
  names within your application
- you need to update your software and configuration to use

After I was very furious about the need to change my software and configuration I now thing, that the access with kernel 3.4.67+ is much saver and the additional work to access the port was just about a hour.
But it was even harder to make sense of the numbers. To get all the information I summarize here

took me almost a complete day. The situation was even worse, because I needed to access the GPIO ports on the LCD connector.

That was actually the motivation to write all of this down, even when some of these information is available elsewhere.

3.2 Kernel 3.3.0+ boot sequence output

This was taken from `/var/log/dmesg`:

```
[    0.244571] [gpio-inf] aw gpio init start
[    0.244603] [gpio-inf] gpio_clk_init: apb pio clk enable success
[    0.244820] gpiochip_add: registered GPIOs 0 to 17 on device: GPA
[    0.244999] gpiochip_add: registered GPIOs 24 to 47 on device: GPB
[    0.245189] gpiochip_add: registered GPIOs 54 to 78 on device: GPC
[    0.245359] gpiochip_add: registered GPIOs 85 to 112 on device: GPD
[    0.245539] gpiochip_add: registered GPIOs 119 to 130 on device: GPE
[    0.245736] gpiochip_add: registered GPIOs 137 to 142 on device: GPF
[    0.245950] gpiochip_add: registered GPIOs 149 to 160 on device: GPG
[    0.246145] gpiochip_add: registered GPIOs 167 to 194 on device: GPH
[    0.246344] gpiochip_add: registered GPIOs 201 to 222 on device: GPI
[    0.246710] [gpio-inf] aw gpio init done
```

This is one of the first actions taken at boot time. So have a look at the beginning of search for “GPIO”.

Under `/sys/class/gpio` all GPIO chip are visable but only with the first port number they are serving.

This is the content of the GPIO directory:

```
ls -l /sys/class/gpio
```

```
total 0
--w------- 1 root root 4096 Sep 27 13:37 export
lrwxrwxrwx 1 root root    0 Sep 27 13:37 gpiochip0 -> ../../devices/virtual/gpio/gpiochip0
lrwxrwxrwx 1 root root    0 Sep 27 13:37 gpiochip119 -> ../../devices/virtual/gpio/gpiochip119
lrwxrwxrwx 1 root root    0 Sep 27 13:37 gpiochip137 -> ../../devices/virtual/gpio/gpiochip137
lrwxrwxrwx 1 root root    0 Sep 27 13:37 gpiochip149 -> ../../devices/virtual/gpio/gpiochip149
lrwxrwxrwx 1 root root    0 Sep 27 13:37 gpiochip167 -> ../../devices/virtual/gpio/gpiochip167
lrwxrwxrwx 1 root root    0 Sep 27 13:37 gpiochip201 -> ../../devices/virtual/gpio/gpiochip201
lrwxrwxrwx 1 root root    0 Sep 27 13:37 gpiochip229 -> ../../devices/virtual/gpio/gpiochip229
lrwxrwxrwx 1 root root    0 Sep 27 13:37 gpiochip24 -> ../../devices/virtual/gpio/gpiochip24
lrwxrwxrwx 1 root root    0 Sep 27 13:37 gpiochip54 -> ../../devices/virtual/gpio/gpiochip54
lrwxrwxrwx 1 root root    0 Sep 27 13:37 gpiochip85 -> ../../devices/virtual/gpio/gpiochip85
--w------- 1 root root 4096 Sep 27 13:37 unexport
```

3.3 Kernel 3.4.67+ boot sequence output

This was taken from `/var/log/dmesg`:

```
...
[    0.364400] sunxi_gpio driver init ver 1.3
[    0.370641] gpiochip_add: registered GPIOs 1 to 75 on device: A1X_GPIO
…
```

This is one of the first actions taken at boot time. So have a look at the beginning of search for “GPIO”.

This leads to the fact, that under `/sys/class/gpio` only one GPIO chip is visable – a little bis misleading!

This is the content of the GPIO directory:

```
ls -l /sys/class/gpio/
```

```
total 0
--w------- 1 root root 4096 Sep 26 10:37 export
lrwxrwxrwx 1 root root    0 Sep 25 02:37 gpio47_ph2 -> ../../devices/platform/gpio-
sunxi/gpio/gpio47_ph2
lrwxrwxrwx 1 root root    0 Sep 26 10:08 gpiochip1 -> ../../devices/platform/gpio-
sunxi/gpio/gpiochip1
--w------- 1 root root 4096 Sep 26 10:08 unexport
```

4. Accessing the GPIO Ports

Just make this short. There are 4 steps you need to follow to access a GPIO port under Linux. Beside of the names of the devices, there are no differences between the different kernels.

Because 3.4.67+ is the newer (and after thinking about it) more straight forward, this example refers to a device named `gpio64_pg0` – the new way!

You can read data from a given GPIO port. I figured out, that 0V for LOW (or 0) and 2.4V for HIGH (or 1) are good numbers. All voltages are measured against ground (GND).

If the input signal is to high, you will at least destroy the port!

Under Linux, the GPIO ports are normal devices read- and write-able with a normal commands. The given examples are using shell commands or C (compiled with gcc).

You will see, that the procedure to read and write data is almost the same. The examples here are using shell commands. If you are interested in an example in C please have a look to Appendix A.

4.1 Read data from a GPIO port

These steps a necessary:

Export the GPIO port to be able to access the port. This just needs to be done at the very beginning and only once. This command will give you the device you can access.

1. export the GPIO port (create the device)

Type: `echo 64 > /sys/class/gpio/export`

Check with: `ls -l /sys/class/gpio`

2. Set the GPIO port to INPUT

Type: `echo in > /sys/class/gpio/gpio64_pg0/direction`
Check with: `cat /sys/class/gpio/gpio64_pg0/direction`

3. Read the current value

Type: `cat /sys/class/gpio/gpio64_pg0/value`

4. unexport the GPIO port (remove the device)

Type: `echo 64 > /sys/class/gpio/unexport`

Check with: `ls -l /sys/class/gpio`

4.2 Write data to a GPIO port

These steps a necessary:

Export the GPIO port to be able to access the port. This just needs to be done at the very beginning and only once. This command will give you the device you can access.

1. export the GPIO port (create the device)

Type: `echo 64 > /sys/class/gpio/export`

Check with: `ls -l /sys/class/gpio`

2. Set the GPIO port to OUTPUT

Type: `echo out > /sys/class/gpio/gpio64_pg0/direction`
Check with: `cat /sys/class/gpio/gpio64_pg0/direction`

3. wite a value

for LOW (0V output):

Type: `echo 0 > /sys/class/gpio/gpio64_pg0/value`

for HIGH (3.3V output):

Type: `echo 1 > /sys/class/gpio/gpio64_pg0/value`

Check with: cat `/sys/class/gpio/gpio64_pg0/value`
and with a DMM at the given physical pin

4. unexport the GPIO port (remove the device)

Type: `echo 64 > /sys/class/gpio/unexport`

Check with: `ls -l /sys/class/gpio`

5. The files `script.bin` and `script.fex`

First of all, the `script.bin` (located in `/boot`) file is a binary files read by the boot loader and the `script.fex` file is the human readable version of it. So, I'll take just about the `script` file.

5.1 Toolset

The two representation of the file can be converted one into the other by using the `sunxi-tools`. The toolset is located in the `/opt/sunxi-tools` directory (the location under kernel 3.3.0+ is `/root/sunxi-tools`).

These are the tools:

`README`	A description of the complete toolset
`bin2fex`	conversion from `.bin` to `.fex`
`fex2bin`	conversion from `.fex` to `.bin`

For more detail information have a look at http://linux-sunxi.org/Main_Page and http://linux-sunxi.org/Sunxi-tools.

5.2 Content of the `script` file

The purpose of this file is to describe the hardware to the kernel and the drivers. So be very careful if you like to change something!

But it's worth looking at the content.

You may want (because you are reading this document) to manipulate the GPIO part. If you change something make sure (and double check) that your change dose to end up in a conflicting situation! Meaning: same ports used for different purposes.

Within the file you can also switch on and off some hardware features. Below, there is an example for i2c port 0 (called twi0 here: tow-wire-interface – the old style to call i2c):

```
[twi0_para]
twi0_used = 1
twi0_scl = port:PB00<2><default><default><default>
twi0_sda = port:PB01<2><default><default><default>
```

So, if you like to disable (for whatever reason) the `twi0`, simply change the second line to

```
twi0_used = 0
```

The result would be simple: the ports PB00 and PB01 would be free for something else!

After you have changed something, you need to generate the `script.bin` file from your newly

created `script.fex` file. The `script.bin` file then needs to be copied to `/boot` (make sure, the directory is mounted!) and reboot your system for the changes take effect!

Here is a very short example how to do it:

Go to the tools directory
```
cd /opt/sunxi-tools
```

Create a new directory
```
mkdir script_mod
```

Enter this directory
```
cd script_mod
```

Mount the boot partition (Because of `udev` the name of your partition may be different. Please have a look at the `/dev` directory and have a look at the symbolic link `/dev/root`).
```
mount /dev/mmcblk0p1 /boot
```

Save a copy of the current file
```
cp /boot/script.bin /boot/script.bin.save
```

Copy the file to the working directory
```
cp /boot/script.bin .
```

Convert the file to human readable format
```
../bin2fex ./script.bin ./script.fex
```

Do your editing

Convert it back to the binary format
```
../fex2bin -v ./script.fex ./script.bin
```

Copy the file to the boot partition
```
cp ./script.bin /boot/
```

Reboot your system
```
init 6
```

A good source of information is also the official documentation from Allwinner. The PDF version can be found here:

https://dl.linux-sunxi.org/allwinner/Configuration_system_and_GPIO_Management_V1.01.pdf

Have a look around, have fun, but be careful!

6. Modifications to use the LCD connector as GPIO

6.1 Preparation

We need to exchange the `script.bin` file in the /boot directory. Just follow the steps below:

0. Log on as root

1. Chance to the tools directory

```
cd /opt/sunxi-tools
```

2. Create a working directory and enter the directory

```
mkdir gpio_mod
cd gpio_mod
```

3. Mount the /boot directory

```
mount /dev/mmcblk0p1 /boot
```

Because of `udev` the name of your partition may be different. Please have a look at the `/dev` directory and have a look at the symbolic link `/dev/root` The directory might already be mounted somewhere (depending on your configuration in the `/etc/fstab` file). You might check this with the following command:

```
mount
```

The output will look like this:

```
/dev/root on / type ext3 (rw,noatime,errors=remount-ro,user_xattr,acl,barrier=1,data=ordered)
devtmpfs on /dev type devtmpfs (rw,relatime,size=415368k,nr_inodes=103842,mode=755)
tmpfs on /run type tmpfs (rw,nosuid,noexec,relatime,size=83096k,mode=755)
tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,size=5120k)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /run/shm type tmpfs (rw,nosuid,nodev,noexec,relatime,size=166180k)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000)
tmpfs on /tmp type tmpfs (rw,relatime)
tmpfs on /var/tmp type tmpfs (rw,relatime)
/dev/mmcblk0p1 on /media/587A-1A07 type vfat
(rw,nosuid,nodev,relatime,uid=1001,gid=1001,fmask=0022,dmask=0077,codepage=cp437,iocharset=ascii,sho
rtname=mixed,showexec,utf8,flush,errors=remount-ro,uhelper=udisks)
/dev/mmcblk0p1 on /boot type vfat
```

Let's assume for the further steps, that you have mounted the partition with the boot loader stuff in the `/boot` directory.

4. Copy some files

Make a copy of the original `script.bin` and check this:

```
cp /boot/script.bin /boot/script.bin.save
```

Check, that it was successful:

```
ls -l /boot

total 4566
drwx------ 2 olimex olimex    2048 Nov  4  2013 back_up
-rw-r--r-- 1 olimex olimex   56736 Sep 25 15:49 script.bin
-rw-r--r-- 1 olimex olimex   56736 Feb  3  2014 script.bin.save
-rw-r--r-- 1 olimex olimex 4558376 Nov  1  2013 uImage
```

Make a copy of this file into the current directory:

```
cp /boot/script.bin .
```

Check, that it was successful:

```
ls -l

total 92
-rw-r--r-- 1 root root 56736 Sep 25 15:48 script.bin
```

5. Convert the `script.bin` file into the readable `script.fex` file

Just type this:

```
../bin2fex ./script.bin ./script.fex
```

The output will look like this:

```
fexc-bin: ./script.bin: version: 0.1.2
fexc-bin: ./script.bin: size: 56736 (85 sections)
```

Check, that the file exists:

```
ls -l

total 92
-rw-r--r-- 1 root root 56736 Sep 25 15:48 script.bin
-rw-r--r-- 1 root root 32109 Sep 25 15:48 script.fex
```

6.2 Disable the LCD display connectivity

Use your favorite text editor (I'm using vi) to do the following modification to the file `script.fex`.

Tow blocks needs to be modified:

The LCD parameter block:

```
[lcd0_para]
lcd_used = 1
lcd_x = 800
lcd_y = 480
...
```

Change the line

```
lcd_used = 1
```

to

```
lcd_used = 0
```

The LCD inizialization block:

```
[disp_init]
disp_init_enable = 1
disp_mode = 0
…
```

Change the line

```
disp_init_enable = 1
```

to

```
disp_init_enable = 0
```

6.3 Register the ports as GPIO

Use your favorite text editor (I'm using vi) to do the following modification to the file `script.fex`.

```
[gpio_para]
gpio_used = 1
gpio_num = 75
gpio_pin_1 = port:PE00<0><default><default><default>
...
```

Modify the block header

Change the line

```
gpio_num = 75
```

to

```
gpio_num = 103
```

Add the following lines at the end of the `[gpio_param]` block:

```
gpio_pin_76 = port:PD17<0><default><default><default>
gpio_pin_77 = port:PD19<0><default><default><default>
gpio_pin_78 = port:PD21<0><default><default><default>
gpio_pin_79 = port:PD23<0><default><default><default>
gpio_pin_80 = port:PD09<0><default><default><default>
gpio_pin_81 = port:PD11<0><default><default><default>
gpio_pin_82 = port:PD13<0><default><default><default>
gpio_pin_83 = port:PD15<0><default><default><default>
gpio_pin_84 = port:PD01<0><default><default><default>
gpio_pin_85 = port:PD03<0><default><default><default>
gpio_pin_86 = port:PD05<0><default><default><default>
gpio_pin_87 = port:PD07<0><default><default><default>
gpio_pin_88 = port:PD27<0><default><default><default>
gpio_pin_89 = port:PD25<0><default><default><default>
gpio_pin_90 = port:PD16<0><default><default><default>
gpio_pin_91 = port:PD18<0><default><default><default>
gpio_pin_92 = port:PD20<0><default><default><default>
gpio_pin_93 = port:PD22<0><default><default><default>
gpio_pin_94 = port:PD08<0><default><default><default>
gpio_pin_95 = port:PD10<0><default><default><default>
gpio_pin_96 = port:PD12<0><default><default><default>
gpio_pin_97 = port:PD14<0><default><default><default>
gpio_pin_98 = port:PD00<0><default><default><default>
gpio_pin_99 = port:PD02<0><default><default><default>
```

```
gpio_pin_100 = port:PD04<0><default><default><default>
gpio_pin_101 = port:PD06<0><default><default><default>
gpio_pin_102 = port:PD26<0><default><default><default>
gpio_pin_103 = port:PD24<0><default><default><default>
```

Add the following lines at the end of the `[gpio_init]` block:

```
pin_76 = port:PD17<0><default><default><default>
pin_77 = port:PD19<0><default><default><default>
pin_78 = port:PD21<0><default><default><default>
pin_79 = port:PD23<0><default><default><default>
pin_80 = port:PD09<0><default><default><default>
pin_81 = port:PD11<0><default><default><default>
pin_82 = port:PD13<0><default><default><default>
pin_83 = port:PD15<0><default><default><default>
pin_84 = port:PD01<0><default><default><default>
pin_85 = port:PD03<0><default><default><default>
pin_86 = port:PD05<0><default><default><default>
pin_87 = port:PD07<0><default><default><default>
pin_88 = port:PD27<0><default><default><default>
pin_89 = port:PD25<0><default><default><default>
pin_90 = port:PD16<0><default><default><default>
pin_91 = port:PD18<0><default><default><default>
pin_92 = port:PD20<0><default><default><default>
pin_93 = port:PD22<0><default><default><default>
pin_94 = port:PD08<0><default><default><default>
pin_95 = port:PD10<0><default><default><default>
pin_96 = port:PD12<0><default><default><default>
pin_97 = port:PD14<0><default><default><default>
pin_98 = port:PD00<0><default><default><default>
pin_99 = port:PD02<0><default><default><default>
pin_100 = port:PD04<0><default><default><default>
pin_101 = port:PD06<0><default><default><default>
pin_102 = port:PD26<0><default><default><default>
pin_103 = port:PD24<0><default><default><default>
```

6.4 Create a new `script.bin` file

Just type this:

```
../fex2bin -v ./script.fex ./script.bin
```

The output will look like this:

```
../fex2bin: from fex:./script.fex to bin:./script.bin
```

Check, that the new file was created:

```
ls -l
```

```
total 92
-rw-r--r-- 1 root root 56736 Sep 26 12:00 script.bin
-rw-r--r-- 1 root root 32109 Sep 26 11:55 script.fex
```

Check the new timestamp!

6.5 Copy the new file to the `/boot` directory and reboot

Just type this:

```
cp ./script.bin /boot/
```

Check, that this was done correctly:

```
ls -l /boot
```

```
total 4566
drwx------ 2 olimex olimex    2048 Nov  4  2013 back_up
-rw-r--r-- 1 olimex olimex   56736 Sep 26 12:05 script.bin
-rw-r--r-- 1 olimex olimex   56736 Feb  3  2014 script.bin.save
-rw-r--r-- 1 olimex olimex 4558376 Nov  1  2013 uImage
```

Check the new timestamp again!

Now, it is time to reboot your system:

```
init 6
```

If you have done your work correctly, the system should come up without any problem!

6.6 Check the new configuration

Now, log on to the system as root and perform some checks.

In the `/var/log/dmesg` (search for GPIO) file you should find something like this:

```
[    0.364420] sunxi_gpio driver init ver 1.3
[    0.371050] gpiochip_add: registered GPIOs 1 to 103 on device: A1X_GPIO
```

Now, check the GPIO directory:

```
ls -l /sys/class/gpio/
```

```
total 0
--w------- 1 root root 4096 Sep 26 10:37 export
lrwxrwxrwx 1 root root    0 Sep 25 02:37 gpio47_ph2 -> ../../devices/platform/gpio-
sunxi/gpio/gpio47_ph2
lrwxrwxrwx 1 root root    0 Sep 26 10:08 gpiochip1 -> ../../devices/platform/gpio-
sunxi/gpio/gpiochip1
--w------- 1 root root 4096 Sep 26 10:08 unexport
```

If you have not modified your configuration for GPIO, only the gpio47_ph2 device should be visable.

Now export one of the newly inserted GPIO devices. We take number 100:

```
echo 100 > /sys/class/gpio/export
```

There should be no error or any other text in return! Now check, that the devise is visable:

```
ls -l /sys/class/gpio/
```

```
total 0
--w------- 1 root root 4096 Sep 26 12:07 export
lrwxrwxrwx 1 root root    0 Sep 25 02:37 gpio47_ph2 -> ../../devices/platform/gpio-
sunxi/gpio/gpio47_ph2
lrwxrwxrwx 1 root root    0 Sep 26 10:37 gpio100_pd4 -> ../../devices/platform/gpio-
sunxi/gpio/gpio100_pd4
lrwxrwxrwx 1 root root    0 Sep 26 10:08 gpiochip1 -> ../../devices/platform/gpio-
sunxi/gpio/gpiochip1
--w------- 1 root root 4096 Sep 26 10:08 unexport
```

6.7 Do you like to check the success on the hardware level?

OK, let's do it!

You definitely need a DMM for this. In addition to simplify the measurement you might used these too:

- a 40-pin ribbon cable connected to the LCD connector
- 2 thin cables

Bring the newly exported GPIO port into the OUTPUT state:

```
echo out > /sys/class/gpio/gpio100_pd4/direction
```

There should be no error!

Check, that the value is 0:

```
cat /sys/class/gpio/gpio100_pd4/value
```

```
0
```

Take your DMM and check the voltage between the following pins:

LCD Pin 2 (GND)
LCD Pin 25 (gpio100_pd4)

The reading should be 0V.

Now switch to HIGH:

```
echo 1 > /sys/class/gpio/gpio75_pg11/value
```

and check the value again:

```
cat /sys/class/gpio/gpio100_pd4/value
```

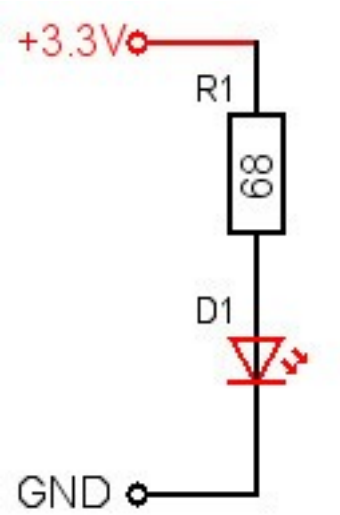
```
0
```

Now, take your DMM again and check the voltage between the same pins:

LCD Pin 2 (GND)
LCD Pin 25 (gpio100_pd4)

The reading should be 3.3V now.

Instead of a DMM, you may use a LED with a 68 Ohm resistor.



Connect resistor to the GPIO port, the longer leg of the LED to the resistor and the shorter leg of the LED to ground (pins 2 and 4 on each GPIO connectors).

If you like, you may exchange the resistor to an 100 Ohm one. You will reduce the current through the LED and prolonger there lifetime.

6.8 Side effects

As I have seen so far, there are no side effects of the modification. Please correct me if I'm wrong!

7. Changing the original Port numbering

I don't think, that this makes sense because the newly introduced numbering of the GPIO ports is logical.

Anyway, if you like to change it, you need to follow the steps to modify the `script.fex` file accordingly.

Just change the numbers in the `[gpio_para]` section of the file and the corresponding lines in the `[gpio_init]` section.

Here is just an example of what you need to do for exchanging GPIO port 1 and 2.

In the `[gpio_para]` section modify:

Original:

```
[gpio_para]
gpio_used = 1
gpio_num = 75
gpio_pin_1 = port:PE00<0><default><default><default>
gpio_pin_2 = port:PE01<0><default><default><default>
…
```

Updated:

```
[gpio_para]
gpio_used = 1
gpio_num = 75
gpio_pin_1 = port:PE01<0><default><default><default>
gpio_pin_2 = port:PE00<0><default><default><default>
…
```

And in the `[gpio_init]` section modify:

Original:

```
[gpio_init]
pin_1 = port:PE00<0><default><default><default>
pin_2 = port:PE01<0><default><default><default>
…
```

Updated:

```
[gpio_init]
pin_1 = port:PE01<0><default><default><default>
pin_2 = port:PE00<0><default><default><default>
…
```

Now generate the `script.bin` file (see last chapter), place it under `/boot` and reboot the system. After this, you have changed the number (but not the port suffix giving the physical pin number) of the device name.

Appendix A: GPIO Test Program for kernel 3.3.0+ and 3.4.67+

This program has been tested under the disc images for kernel 3.3.0+ and 3.4.67+. You might find the program and the functions useful. Feel free to use the code, but report suggestions and errors to me!

If you find this program “dirty” - it is! For a better understanding, there is no error handling included! If you like to use this code (or part of it) in production, you need to add a sophisticated error handling because this will be the foundation of your application!

A20-GPIO-test.c

```
/****************************************************************************/
/**                                                                        **/
/**  A20-GPIO-test.c                                                       **/
/**                                                                        **/
/**  This program is just for test and demonstration purposes!             **/
/**  It comes with no warrenty of any kind, no error handling, but         **/
/**  in the hope, that it will be useful.                                  **/
/**                                                                        **/
/**  This program is free software. You may change it, re-distribute,      **/
/**  and use it as long as you leave the authors data unchanged.           **/
/**                                                                        **/
/**  Author: Dr. Guido Pelz (DM1GP / PY6ZGP)                               **/
/**          www.dm1gp.com and www.py6zgp.com                              **/
/**                                                                        **/
/**  History: 27.9.2014  (initial version)                                 **/
/**                                                                        **/
/****************************************************************************/


#include <string.h>
#include <stdio.h>
#include <errno.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <unistd.h>
#include <dirent.h>


#define TRUE                            1
#define FALSE                           0

#define WAIT_TIME_SEC                   1

#define A20GPIO_MODE_INPUT              1
#define A20GPIO_MODE_OUTPUT             2

#define MAX_PORT_FILE_LENGTH          255


/* two defines for the port direction */
enum {
  CHANNEL_TYPE_OUTPUT,
  CHANNEL_TYPE_INPUT,};


/* the structure for the port information */
typedef struct {
  int PortNo;
  int IOMode;
  char PortFile[MAX_PORT_FILE_LENGTH];
  char PortDirectionFile[MAX_PORT_FILE_LENGTH];
  char PortValueFile[MAX_PORT_FILE_LENGTH];
```

```
} Port_t;


/* The global structure for a port */
Port_t Port;



/*****************************************************************************/
/**                                                                         **/
/**  Read the complete name of the device directory                         **/
/**                                                                         **/
/**  Output:   PortFileName: the name of the directory for the port         **/
/**  Return:   -errno, if an error has occured, otherwise 0                 **/
/**                                                                         **/
/*****************************************************************************/

int GetPortFileName (char *PortFileName) {

  DIR *dirp;
  struct dirent *dp;
  char dirname[MAX_PORT_FILE_LENGTH], item[MAX_PORT_FILE_LENGTH];

  /* This is the pattern for the GPIO file name */
  sprintf(item, "gpio%i", Port.PortNo);

  PortFileName[0] = 0;

  /* This is the GPIO file directory */
  sprintf(dirname, "/sys/class/gpio");
  /* open the directory */
  dirp = opendir(dirname);
  /* Check if the directory could be opened for read */
  if (dirp != NULL) {
    /* now check all files in the directory */
    while ((dp = readdir(dirp)) != NULL) {
      /* if found, return the complete file name with directory */
      if (strncmp(dp->d_name, item, strlen(item)) == 0) {
        sprintf(PortFileName, "%s/%s", dirname, dp->d_name);
      }
    }
    /* close */
    closedir(dirp);
  }

  return(errno);
}




/*****************************************************************************/
/**                                                                         **/
/**  Export the GPIO port to add the port to the                            **/
/**  /sys/call/gpio file system and make it available for further use.      **/
/**                                                                         **/
/**  Return:   -errno, if an error has occured, otherwise 0                 **/
/**                                                                         **/
/*****************************************************************************/

int A20GPIO_ExportGPIOPort () {

  FILE *fh_export;

  printf("Export port no. %i ... ", Port.PortNo);

  /* open the export file */
  fh_export = fopen("/sys/class/gpio/export", "w");
  if (fh_export != NULL) {
    /* write the number of the port to be exported to the file */
    fprintf(fh_export, "%i\n", Port.PortNo);
    /* close the export file */
```

```
    fclose(fh_export);
  }

  /* print an error message, if aplicable */
  if (errno != FALSE) {
    printf("Error: %i - %s\n", errno, strerror(errno));
  } else {
    printf("done\n");
  }

  return(-errno);
}
```

```
/*****************************************************************************/
/**                                                                         **/
/**  Unexport the GPIO port to remove the port from                         **/
/**  the /sys/call/gpio file system                                         **/
/**                                                                         **/
/**  Return:   -errno, if an error has occured, otherwise 0                 **/
/**                                                                         **/
/*****************************************************************************/

int A20GPIO_UnExportGPIOPort () {

  FILE *fh_export;

  printf("Unexport port no. %i ... ", Port.PortNo);

  /* open the unexport file */
  fh_export=fopen("/sys/class/gpio/unexport", "w");
  if (fh_export != NULL) {
    /* write the number of the port to be unexported to the file */
    fprintf(fh_export, "%i\n", Port.PortNo);
    /* close the unexport file */
    fclose(fh_export);
    printf("done\n");
  }

  /* print an error message, if aplicable */
  if (errno != FALSE) printf("Error: %i - %s\n", errno, strerror(errno));

  return(-errno);
}
```

```
/*****************************************************************************/
/**                                                                         **/
/**  Sets the mode of the given port to INPUT or OUTPUT.                    **/
/**                                                                         **/
/**  Input:  IOMode:   - the mode of the port: 1 - INPUT, 2 - OUTPUT        **/
/**  Return:   -errno, if an error has occured, otherwise 0                 **/
/**                                                                         **/
/*****************************************************************************/

int A20GPIO_pinMode (int IOMode) {

  FILE *fh_direction;

  printf("Set IO mode for port to %i ... ", IOMode);
  /* open the direction file */
  fh_direction = fopen(Port.PortDirectionFile, "w");
  /* write "in" or "out" to the file */
  if (fh_direction != NULL) {
    switch (IOMode) {
      case CHANNEL_TYPE_INPUT: fprintf(fh_direction, "in");
                               break;
```

```
      case CHANNEL_TYPE_OUTPUT:fprintf(fh_direction, "out");
                               break;
      default:                 errno = EINVAL;
                               break;
    }
    /* close the direction file */
    fclose(fh_direction);
    printf("done!\n");
  }

  /* print an error message, if aplicable */
  if (errno != FALSE) printf("Error: %i - %s\n", errno, strerror(errno));

  return(-errno);
}
```

```
/*************************************************************************/
/**                                                                     **/
/**  Read the current value of a port. Can be used for ports set to     **/
/**  INPUT or OUTPUT.                                                   **/
/**                                                                     **/
/**  Output:  RawValue:   - the current value of the port: 0 or 1       **/
/**  Return:   -errno, if an error has occured, otherwise 0             **/
/**                                                                     **/
/*************************************************************************/

int A20GPIO_ReadValue(int *RawValue) {

  FILE *fh_port;

  printf("Read state of port ... ");
  /* open the value file */
  fh_port = fopen(Port.PortValueFile, "r");
  /* read one character from the file */
  if (fh_port != NULL) {
    /* read one character from the file.                          */
    /* We like to have a number not a character. So substract     */
    /* 48, because the ASCII code for 0 is 48 and for 1 it is 49. */
    *RawValue = fgetc(fh_port) - 48;
    /* close the value file */
    fclose(fh_port);
    printf("done, state = %i!\n", *RawValue);
  }

  /* print an error message, if applicable */
  if (errno != FALSE) printf("Error: %i - %s\n", errno, strerror(errno));

  return(-errno);
}
```

```
/*************************************************************************/
/**                                                                     **/
/**  Write a value to a port in OUTPUT mode.                            **/
/**                                                                     **/
/**  Input:  RawValue:   - the new value of the port: 0 or 1            **/
/**  Return:   -errno, if an error has occured, otherwise 0             **/
/**                                                                     **/
/*************************************************************************/

int A20GPIO_WriteValue(int RawValue) {

  FILE *fh_port;

  printf("Set state for port to %i ... ", RawValue);
  /* open the value file */
```

```
  fh_port=fopen(Port.PortValueFile, "w");
  if (fh_port != NULL) {
    /* write a 0 or 1 as a character to the file */
    fprintf(fh_port, "%i\n", RawValue);
    /* close the value file */
    fclose(fh_port);
    printf("done!\n");
  }

  /* print an error message, if applicable */
  if (errno != FALSE) printf("Error: %i - %s\n", errno, strerror(errno));

  return(-errno);
}




/****************************************************************************/
/**                                                                        **/
/**  The main program.                                                     **/
/**                                                                        **/
/****************************************************************************/

int main (int argc, char *argv[]) {

  char PortFileName[255];
  int retval;

  /* check for command line parameters */
  Port.PortNo = 0;
  if (argc == 2) Port.PortNo = atoi(argv[1]);
  if ((argc != 2) || (Port.PortNo == 0)) {
    printf("Usage: %s <Port Number>\n", argv[0]);
    return(1);
  }


  /* Export the given port: create the device */
  A20GPIO_ExportGPIOPort();

  /* construct the file names for the direction and for the value */
  retval = GetPortFileName(PortFileName);
  sprintf(Port.PortFile, PortFileName);
  sprintf(Port.PortDirectionFile, "%s/direction", Port.PortFile);
  sprintf(Port.PortValueFile, "%s/value", Port.PortFile);

  /* Set the mode to OUTPUT */
  A20GPIO_pinMode(CHANNEL_TYPE_OUTPUT);

  /* Set the pin to HIGH */
  A20GPIO_WriteValue(1);

  /* read the current value */
  A20GPIO_ReadValue(&retval);
  printf("The current value of port no %i is %i.\n", Port.PortNo, retval);

  printf("Wait %i sec ... \n", WAIT_TIME_SEC);
  sleep(WAIT_TIME_SEC);
  printf("done!\n");

  /* Set the pin to LOW */
  A20GPIO_WriteValue(0);

  /* read the current value */
  A20GPIO_ReadValue(&retval);
  printf("The current value of port no %i is %i.\n", Port.PortNo, retval);

  /* Unexport the given port: destroy the device */
  A20GPIO_UnExportGPIOPort();

  printf("Finished!\n");

  return(0);
}
```

```
/*****************************************************************************/
/**                                                                         **/
/**  End of file.                                                           **/
/**                                                                         **/
/*****************************************************************************/
```

To compile this program you need no extra libraries. Just type

```
gcc A20-GPIO-test.c -o A20-GPIO-test -Wall
```

The program should compile with no error messages!

Usage: `./A20-GPIO-test <Port Number>`

This program performs the following tasks for a given port:

1. Export the given port: create the device
2. Gets the file name of the port directory and construct the file names for the ports direction and for the ports value
3. Set the mode to OUTPUT
4. Set the pin to HIGH
5. read the current value
6. Waits a second
7. read the current value
8. Unexport the given port: destroy the device

Appendix B: GPIO Port Summary

Here you can find a comprehensive list of all ports including these purpose.

The columns “used for”, “2nd use”, “3rd use”, and “not used but mentioned” refer to the file `scripts.fex` included in the official image.

It looks like, that within this file there are a couple of inconsistencies. We may sort these out in the future.

B.1 GPIO Chip PA Ports

Port	old Port Number	Linux Device Name	Connector	Pin	used for	not used but metioned in scripts.fex file		
0	0				emac			
1	1				emac			
2	2				emac			
3	3				emac			
4	4				emac			
5	5				emac	spi3		
6	6				emac	spi3		
7	7				emac	spi3		
8	8				emac	spi3		
9	9				emac	spi3		pcm
10	10				emac	uart1		
11	11				emac	uart1		
12	12				emac	uart1		
13	13				emac	uart1		
14	14				emac	uart1		pcm
15	15				emac	uart1		pcm
16	16				emac	uart1	can	pcm
17	17				emac	uart1	can	pcm

emac	Ethernet Media Access Controller
spi	SPI interface
uart	UART interface
can	CAN-Bus interface
pcm	PCM digital audio

B.2 GPIO Chip PB Ports

Port	old Port Number	Linux Device Name	Connector	Pin	used for	2nd use	not used but metioned in scripts.fex file	
0	24		GPIO-2	5	twi0		twi	
1	25		GPIO-2	7	twi0		twi	
2	26		LCD	36	lcd			
3	27	gpio28_pb3	LCD	33		gpio	motor	
			GPIO-3	6				
4	28	gpio29_pb4	LCD	34		gpio	ir	
			GPIO-3	8				
5	29	gpio30_pb5	GPIO-3	10	wifi	gpio	bt	i2s
6	30	gpio31_pb6	GPIO-3	12		gpio		i2s
7	31	gpio32_pb7	GPIO-3	14		gpio		i2s
8	32	gpio33_pb8	GPIO-3	16	sata	gpio		i2s
9	33				usbc0			
10	34	gpio34_pb10	GPIO-3	18		gpio		
11	35	gpio35_pb11	GPIO-3	20		gpio		
12	36	gpio36_pb12	GPIO-3	22		gpio		i2s
13	37	gpio37_pb13	GPIO-3	24		gpio	ctp	spdif
14	38	gpio38_pb14	GPIO-3	26	jtag	gpio		
15	39	gpio39_pb15	GPIO-3	28	jtag	gpio		
16	40	gpio40_pb16	GPIO-3	30	jtag	gpio		
17	41	gpio41_pb17	GPIO-3	32	jtag	gpio		
18	42		UEXT-2	5	twi1			
19	43		UEXT-2	6	twi1			
20	44		UEXT-1	5	twi2			
21	45		UEXT-1	6	twi2			
22	46			36	uart0		uart	uart_debug
23	47			34	uart0		uart	uart_debug

twi	Two Wire Interface (I2C)
lcd	LCD display
wifi	WIFI interface
sata	SATA interface
usb	USB interface
jtag	JTAG bus interface
uart	UART interface
ir	Infrared interface
bt	Bluetoth interface
ctp	Computer-to-Plate?
i2s	Integrated Interchip Sound
spdif	SPDIF digital audio
motor	Motor interface?

For gpio28_pb2 and gpio29_pb4 see comment on the LCD connector (Appendix C.4).

B.3 GPIO Chip PC Ports

Port	old Port Number	Linux Device Name	Connector	Pin	used for	2nd use	3rd use	not used but metioned in scripts.fex file
0	54					nand		
1	55					nand		
2	56					nand		
3	57	gpio21_pc3	GPIO-2	21	gpio			
4	58					nand		
5	59					nand		
6	60					nand	card2	mmc2
7	61	gpio22_pc7	GPIO-2	23	gpio	nand	card2	mmc2
8	62					nand	card2	mmc2
9	63					nand	card2	mmc2
10	64					nand	card2	mmc2
11	65					nand	card2	mmc2
12	66					nand		
13	67					nand		
14	68					nand		
15	69					nand		
16	70	gpio23_pc16	GPIO-2	25	gpio			
17	71	gpio24_pc17	GPIO-2	27	gpio			
18	72	gpio25_pc18	GPIO-2	29	gpio			
19	73		UEXT-1	10	spi2			
20	74		UEXT-1	9	spi2			
21	75		UEXT-1	8	spi2			
22	76		UEXT-1	7	spi2			gps
23	77	gpio26_pc23	GPIO-2	31	gpio			
24	78	gpio27_pc24	GPIO-2	33	gpio			

gpio	General Purpose I/O
spi	SPI interface
nand	NAND flash memory
card	HDD interface?
mmc	Multi Media Card interface

B.4 GPIO Chip PD Ports

Port	old Port Number	Linux Device Name	Connector	Pin	used for
0	85	gpio98_pd0	LCD	21	lcd
1	86	gpio84_pd1	LCD	22	lcd
2	87	gpio99_pd2	LCD	23	lcd
3	88	gpio85_pd3	LCD	24	lcd
4	89	gpio100_pd4	LCD	25	lcd
5	90	gpio86_pd5	LCD	26	lcd
6	91	gpio101_pd6	LCD	27	lcd
7	92	gpio87_pd7	LCD	28	lcd
8	93	gpio94_pd8	LCD	13	lcd
9	94	gpio80_pd9	LCD	14	lcd
10	95	gpio95_pd10	LCD	15	lcd
11	96	gpio81_pd11	LCD	16	lcd
12	97	gpio96_pd12	LCD	17	lcd
13	98	gpio82_pd13	LCD	18	lcd
14	99	gpio97_pd14	LCD	19	lcd
15	100	gpio83_pd15	LCD	20	lcd
16	101	gpio90_pd16	LCD	5	lcd
17	102	gpio76_pd17	LCD	6	lcd
18	103	gpio91_pd18	LCD	7	lcd
19	104	gpio77_pd19	LCD	8	lcd
20	105	gpio92_pd20	LCD	9	lcd
21	106	gpio78_pd21	LCD	10	lcd
22	107	gpio93_pd22	LCD	11	lcd
23	108	gpio79_pd23	LCD	12	lcd
24	109	gpio103_pd24	LCD	31	lcd
25	110	gpio89_pd25	LCD	32	lcd
26	111	gpio102_pd26	LCD	29	lcd
27	112		LCD	30	lcd

lcd LCD display

B.5 GPIO Chip PE Ports

Port	old Port Number	Linux Device Name	Connector	Pin	used for	not used but metioned in scripts.fex file
0	119	gpio1_pe0	GPIO-2	6	gpio	csi0
1	120	gpio2_pe1	GPIO-2	8	gpio	csi0
2	121	gpio3_pe2	GPIO-2	10	gpio	csi0
3	122	gpio4_pe3	GPIO-2	12	gpio	csi0
4	123	gpio5_pe4	GPIO-2	14	gpio	csi0
5	124	gpio6_pe5	GPIO-2	16	gpio	csi0
6	125	gpio7_pe6	GPIO-2	18	gpio	csi0
7	126	gpio8_pe7	GPIO-2	20	gpio	csi0
8	127	gpio9_pe8	GPIO-2	22	gpio	csi0
9	128	gpio10_pe9	GPIO-2	24	gpio	csi0
10	129	gpio11_pe10	GPIO-2	26	gpio	csi0
11	130	gpio12_pe11	GPIO-2	28	gpio	csi0

gpio	General Purpose I/O
csi	Camera Serial Interface

B.6 GPIO Chip PF Ports

Port	old Port Number	Linux Device Name	Connector	Pin	used for	not used but metioned in scripts.fex file
0	137				mmc0	card0
1	138				mmc0	card0
2	139				mmc0	card0
3	140				mmc0	card0
4	141				mmc0	card0
5	142				mmc0	card0

mmc Multi Media Card controller
card HDD interface?

B.7 GPIO Chip PG Ports

Port	old Port Number	Linux Device Name	Connector	Pin	used for	not used but metioned in scripts.fex file		
0	149	gpio64_pg0	GPIO-1	5	gpio	csi1		
1	150	gpio65_pg1	GPIO-1	7	gpio	csi1		
2	151	gpio66_pg2	GPIO-1	9	gpio	csi1		
3	152	gpio67_pg3	GPIO-1	11	gpio	csi1	mmc1	
4	153	gpio68_pg4	GPIO-1	13	gpio	csi1	mmc1	
5	154	gpio69_pg5	GPIO-1	15	gpio	csi1	mmc1	
6	155	gpio70_pg6	GPIO-1	17	gpio	csi1	mmc1	
7	156	gpio71_pg7	GPIO-1	19	gpio	csi1	mmc1	
8	157	gpio72_pg8	GPIO-1	21	gpio	csi1	mmc1	
9	158	gpio73_pg9	GPIO-1	23	gpio	csi1		
10	159	gpio74_pg10	GPIO-1	25	gpio	csi1		
11	160	gpio75_pg11	GPIO-1	27	gpio	csi1		
13							mmc1	(error in script.fex?)

gpio General Purpuse I/O
mmc Multi Media Card
csi Camera Serial Interface

B.8 GPIO Chip PH Ports

Port	old Port Number	Linux Device Name	Connector	Pin	used for	2nd use	not used but metioned in scripts.fex file			
0	167	gpio46_ph0	GPIO-3	5	mmc0	gpio	uart3	lcd1		
1	168					gpio	uart3	lcd1		
2	169	gpio47_ph2	GPIO-3	7	pmu	gpio	uart3	lcd1		
3	170				usbc2		uart3	lcd1		
4	171				usbc0		uart4	lcd1		
5	172				usbc0		uart4	lcd1		
6	173				usbc1		uart6	lcd1	ms	
7	174	gpio48_ph7	GPIO-3	9	lcd0	gpio	uart6	lcd1	ms	
8	175		LCD	35	lcd0			lcd1	ms	kp
9	176	gpio49_ph9	GPIO-3	11	wifi	gpio		lcd1	ms	kp
10	177	gpio50_ph10	GPIO-3	13	wifi	gpio		lcd1	ms	kp
11	178	gpio51_ph11	GPIO-3	15	mmc3	gpio		lcd1	ms	kp
12	179	gpio52_ph12	GPIO-3	17		gpio		lcd1		
13	180	gpio53_ph13	GPIO-3	19		gpio	csi0	lcd1	smc	
14	181	gpio54_ph14	GPIO-3	21		gpio	csi1	lcd1	smc	kp
15	182	gpio55_ph15	GPIO-3	23	audio	gpio		lcd1	smc	kp
16	183	gpio56_ph16	GPIO-3	25		gpio	csi0	lcd1	smc	kp
17	184	gpio57_ph17	GPIO-3	27		gpio	csi1	lcd1	smc	kp
18	185	gpio58_ph18	GPIO-3	29		gpio	csi0	lcd1	smc	kp gy
19	186	gpio59_ph19	GPIO-3	31		gpio		lcd1	smc	kp gy
20	187	gpio60_ph20	GPIO-3	33		gpio		lcd1		ls
21	188	gpio61_ph21	GPIO-3	35		gpio	ctp	lcd1		
22	189	gpio62_ph22	GPIO-3	37		gpio			kp	
23	190	gpio63_ph23	GPIO-3	39		gpio			kp	
24	191	gpio42_ph24	GPIO-3	34		gpio			kp	
25	192	gpio43_ph25	GPIO-3	36		gpio			kp	
26	193	gpio44_ph26	GPIO-3	38		gpio			kp	
27	194	gpio45_ph27	GPIO-3	40		gpio			kp	

mmc	Multi Media Card
pmu	Power Management Unit
usb	USB interface
lcd	LCD display
wifi	Wireless LAN
audio	Audo controller?
gpio	General Purpose I/O
uart	UART interface
csi	Camera Serial Interface
ctp	Computer-to-Plate?
ms	?
smc	SMC devices?
kp	Keypad interface

B.9 GPIO Chip PI Ports

Port	old Port Number	Linux Device Name	Connector	Pin	used for	2nd use	not used but metioned in scripts.fex file		
0	201	gpio15_pi0	GPIO-2	9		gpio			gps
1	202	gpio16_pi1	GPIO-2	11		gpio			gps
2	203	gpio17_pi2	GPIO-2	13		gpio			gps
3	204	gpio18_pi3	GPIO-2	15		gpio	lcd1		
4	205				mmc3				
5	206				mmc3				
6	207				mmc3				
7	208				mmc3				
8	209				mmc3				
9	210				mmc3				
10	211	gpio19_pi10	GPIO-2	17		gpio	spi0		
11	212	gpio20_pi11	GPIO-2	19		gpio	spi0		
12	213		UEXT-1	3	uart6		spi0		
13	214		UEXT-1	4	uart6		spi0	tkey	compass
14	215	gpio13_pi14	GPIO-2	30		gpio	spi0	Ps2_1	gps
15	216	gpio14_pi15	GPIO-2	32		gpio		Ps2_1	gps
16	217		UEXT-2	10	spi1		uart2		
17	218		UEXT-2	9	spi1		uart2		
18	219		UEXT-2	8	spi1		uart2		
19	220		UEXT-2	7	spi1		uart2		
20	221		UEXT-2	3	uart7	wifi	Ps2_0	bt	
21	222		UEXT-2	4	uart7	wifi	Ps2_0	bt	

mmc	Multi Media Card
uart	UART interface
spi	SPI interface
gpio	General Purpose I/O
wifi	Wireless LAN
lcd	LCD Display
ps2	PS2 interface
tkey	Touchkey interface?
bt	Bluetooth interface
gps	GPS device
compa	Compass device

Appendix C: Connector Summary

C.1 GPIO-1

Only the GND, voltages and the GPIO devices names are given!

	Connector Pin		
5V	**1**	**2**	GND
3.3V	**3**	**4**	GND
gpio64_pg0	**5**	**6**	
gpio65_pg1	**7**	**8**	
gpio66_pg2	**9**	**10**	
gpio67_pg3	**11**	**12**	
gpio68_pg4	**13**	**14**	
gpio69_pg5	**15**	**16**	
gpio70_pg6	**17**	**18**	
gpio71_pg7	**19**	**20**	
gpio72_pg8	**21**	**22**	
gpio73_pg9	**23**	**24**	
gpio74_pg10	**25**	**26**	
gpio75_pg11	**27**	**28**	
	29	**30**	
	31	**32**	
	33	**34**	
	35	**36**	
	37	**38**	
	39	**40**	

green: position and function confirmed
blue: see comment below
red: not confirmed

C.2 GPIO-2

Only the GND, voltages and the GPIO devices names are given!

	Connector Pin		
5V	**1**	**2**	GND
3.3V	**3**	**4**	GND
	5	**6**	gpio1_pe0
	7	**8**	gpio2_pe1
gpio15_pi0	**9**	**10**	gpio3_pe2
gpio16_pi1	**11**	**12**	gpio4_pe3
gpio17_pi2	**13**	**14**	gpio5_pe4
gpio18_pi3	**15**	**16**	gpio6_pe5
gpio19_pi10	**17**	**18**	gpio7_pe6
gpio20_pi11	**19**	**20**	gpio8_pe7
gpio21_pc3	**21**	**22**	gpio9_pe8
gpio22_pc7	**23**	**24**	gpio10_pe9
gpio23_pc16	**25**	**26**	gpio11_pe10
gpio24_pc17	**27**	**28**	gpio12_pe11
gpio25_pc18	**29**	**30**	gpio13_pi14
gpio26_pc23	**31**	**32**	gpio14_pi15
gpio27_pc24	**33**	**34**	
	35	**36**	
	37	**38**	
	39	**40**	

green: position and function confirmed
blue: see comment below
red: not confirmed

C.3 GPIO-3

Only the GND, voltages and the GPIO devices names are given!

	Connector Pin		
5V	**1**	**2**	GND
3.3V	**3**	**4**	GND
gpio46_ph0	**5**	**6**	gpio28_pb3
gpio47_ph2	**7**	**8**	gpio29_pb4
gpio48_ph7	**9**	**10**	gpio30_pb5
gpio49_ph9	**11**	**12**	gpio31_pb6
gpio50_ph10	**13**	**14**	gpio32_pb7
gpio51_ph11	**15**	**16**	gpio33_pb8
gpio52_ph12	**17**	**18**	gpio34_pb10
gpio53_ph13	**19**	**20**	gpio35_pb11
gpio54_ph14	**21**	**22**	gpio36_pb12
gpio55_ph15	**23**	**24**	gpio37_pb13
gpio56_ph16	**25**	**26**	gpio38_pb14
gpio57_ph17	**27**	**28**	gpio39_pb15
gpio58_ph18	**29**	**30**	gpio40_pb16
gpio59_ph19	**31**	**32**	gpio41_pb17
gpio60_ph20	**33**	**34**	gpio42_ph24
gpio61_ph21	**35**	**36**	gpio43_ph25
gpio62_ph22	**37**	**38**	gpio44_ph26
gpio63_ph23	**39**	**40**	gpio45_ph27

green: position and function confirmed
blue: see comment below
red: not confirmed

gpio47_ph2 is used for the green on-board LED (blinking)
Better not using this port.

C.4 LCD

Only the GND, voltages and the GPIO devices names are given!

The devices with higher numbers than 75 can only be used with the mentioned modifications of the script.fex file!

	Connector Pin		
5V	1	2	GND
3.3V	3	4	GND
gpio90_pd16	5	6	gpio76_pd17
gpio91_pd18	7	8	gpio77_pd19
gpio92_pd20	9	10	gpio78_pd21
gpio93_pd22	11	12	gpio79_pd23
gpio94_pd8	13	14	gpio80_pd9
gpio95_pd10	15	16	gpio81_pd11
gpio96_pd12	17	18	gpio82_pd13
gpio97_pd14	19	20	gpio83_pd15
gpio98_pd0	21	22	gpio84_pd1
gpio99_pd2	23	24	gpio85_pd3
gpio100_pd4	25	26	gpio86_pd5
gpio101_pd6	27	28	gpio87_pd7
gpio102_pd26	29	30	gpio88_pd27
gpio103_pd24	31	32	gpio89_pd25
gpio28_pb3	33	34	gpio29_pb4
	35	36	
	37	38	
	39	40	

green: position and function confirmed
blue: see comment below
red: not confirmed

Because the positions of gpio28_pb3 and gpio29_pb4 could not be verified, I assume, that the official port map (picture in chapter 2) is wrong. Another strong hint is, that these two ports are working on the GPIO-3 connector. From my point of view, it makes no sense to map a single port to two connectors.