# OpenBTS for dummies - v0.5

Axelle Apvrille, Fortinet
aapvrille@fortinet.com

April 19, 2011

**Abstract**

This document is to be seen as a guideline or a collection of notes for newbies to OpenBTS who struggle to get it working, or are lost in the wiki pages and wonder where to start.

Mostly, I detail here how I got it to work on my side, from step to step, with answers I found to a few issues I faced or my understanding of the problem.

*Please feel free to send in corrections as I am not an OpenBTS expert.*

I would certainly recommend reading [Ale09, oped, opec], documents that I found very useful. The OpenBTS mailing-list [opee] is very responsive too.

## Contents

## 1 What is OpenBTS? (brief)

Litteraly, OpenBTS is an *open Base Transceiver Station*, where a BTS is the telecom equipment the closesest to the mobile phone. See the nice GSM network diagram in [Ale09].

On an end-user point of view, with OpenBTS, GSM phones can call each other, send SMS to each other etc. From an inner perspective, OpenBTS's Wikipedia page [opeb] explains it quite well:

> *"OpenBTS replaces the traditional GSM operator network switching subsystem infrastructure, from the Base Transceiver Station (BTS) upwards. Instead of forwarding call traffic through to an operator's mobile switching centre (MSC) the calls are terminated on the same box by forwarding the data onto the Asterisk PBX via SIP and Voice-over-IP (VoIP)."*

From an administrator's point of view, OpenBTS consists of a Universal Software Radio Peripheral (USRP) board, connected on a USB port of a Linux box running Asterisk, GnuRadio and OpenBTS.

## 2 Hardware Requirements

This is what you need to get OpenBTS up and running. Prices are approximate.

- **Computer**. Basically, any computer should do the work. The only thing which is really required is a USB port to plug the USRP board, but all computers usually have that...

Figure 1: SIM Max card

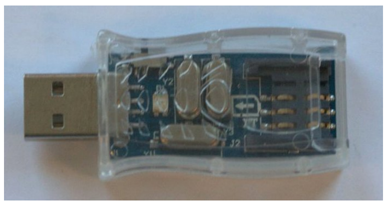


Figure 2: SIM card reader/writer

- **USRP 1**. This board (see Figure 3) can be purchased from Ettus Research (http://www.ettus.com/) for 700 USD. OpenBTS should soon support USRP 2. See the mailing-list [opee] for news concerning this.
- **Daughterboard(s)**. I use a single daughterboard, but for coverage and quality of signals, it is better to use 2 daughterboards. Select the daughterboard you need according to the GSM band you want to use. RFX 900 for GSM 850/900, RFX 1800 for GSM 1800/1900. Price from Ettus: 275 USD. Actually, you might consider buying a RFX1800 in all cases, because it is easy to change a RFX1800 into a RFX900 (firmware flash - no hardware modification) whereas changing a RFX900 into a RFX1800 requires a hardware modication (removing an ISM filter).
- **Antenna**. 1 antenna per daughterboard. Be sure to select an antenna that matches your daughterboard. Can be purchased from Ettus for 35 USD.
- **52 Mhz clock**. Most of the time, this is required but check out Section 4 for more information.
- **Mobile phones**. Obviously you need one at least. It must be *unlocked*. And you need to be able to manually select a network for that phone (see Figure 12).
- **SIM cards**. ... and of course, one SIM card per mobile phone. It is possible to use a standard SIM card - the one you use in your own mobile phone[1], or you can buy a programmable SIM card (see Figure 1). Search for something like Super SIM, SIM MAX, Magic SIM on the web. For each SIM card, you need to know its IMSI (section 7.2 explains how to get it).
- **Magic SIM card reader/writer**. If you use Magic SIM cards, you need to card reader and writer to program the SIM card (see Figure 2). This usually costs only a few bucks.

My personal configuration is listed at Table 1.

## 3 From the USRP kit to the USRP - newbies only

The USRP usually ships in a ”kit”, but it is very easy to mount. Just need a screwdriver. You can do it, even if you are a hardware dummy :)

1. screw the mainboard onto the black enclosure
2. on the mainboard, screw the special screws which make sure the daughterboards are elevated just right above the mainboard

[1]no, it won’t ruin it. But backup your SIM contacts and SMS.

| Type | Specifications |
|---|---|
| Computer | Dell Optiplex 170L, with a 2.4 Ghz processor and 1 Go RAM |
| USRP | USRP 1 Rev 4.5 board - bought from `www.ettus.com` |
| Daughterboard | 1 RFX 1800, 1.5-2.1 GHz Transceiver, 100+mW output. |
| Antenna | VERT 900, 824-960 MHz, 1710-1990 MHz Quad-band Cellular/PCS and ISM Band Vertical Antenna, 3dBi Gain, 9 Inches, Works with WBX, RFX900, RFX1800 |
| Clock | TXCO 52.00Mhz, frequency stability: ± (max) ppm, aging 1 (max) ppm, operational temperature; -20 to 70 °C. Costs approximately 13 USD |
| SIM Card Reader/Writer | 12in1 SIM Card USB Card Reader/Writer Copier Cloner GSM. Bought from `www.1powershop.com` for approximately 14 USD |
| 2 Mobile phones | Nokia 6680 and Nokia N95 |

Table 1: Hardware specifications

3. connect the RFX daughterboard. If a single board, make sure to connect it on side A (notice the words RXA and TXA) on the right of the board when you are facing the USRP.
4. install and connect the ventilator to the motherboard
5. screw the RF cables on the daughterboard and have them go to the enclosure's front panel.
6. screw the antenna to the RF cable that matches TX/RX of the daughterboard. The other RF cable is unused in my case
7. close the enclosure (actually, I suggest to leave it open to be able to check everything is connected fine)
8. use the USB cable to connect the USRP to your computer
9. connect the power supply cable to power

Have a look at Ettus's web site to see what the USRP looks like in the end (Figure 3).

# 4 Clocks

## 4.1 Do I really need another clock?

The USRP board ships with a 64Mhz clock, but [oped] explains why it is insufficient in most situations. Basically, unless you are using RFX900 and are very lucky, you will require a better clock than the standard one that ships with the USRP board. If you are short of money, you can give it a try, but if your phones don't register in the end, too bad, you'll have to open the piggy bank.

## 4.2 52Mhz clocks

The possible solutions are:

- at one time, Kestrel Signal Processing (`http://kestrelsignalprocessing.mybigcommerce.com/categories/OpenBTS-Hardware/`) used to sell a clock for 150 USD. It's now out of stock.
- FairWave's Clock Tamer (`http://www.fairwaves.ru/`) has a clock for 250 USD. It is also sold out, but should be soon back in stock.

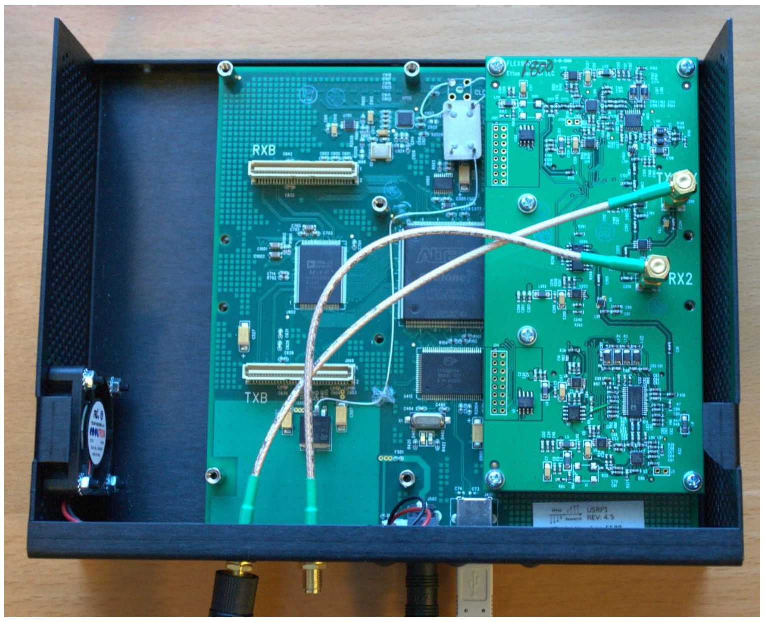


Figure 3: USRP 1 Rev 4.5 with additional 52Mhz clock

- FA-SY1 (`http://www.box73.de/catalog/product_info.php?\-products_id=1869&osCsid=omhrb00t7pga6lqnbq7q3u6ak0`) can be bought for 39 euros. The price is attractive but the voltage output is too high (3.3V whereas the distribution chip only copes up to 2V max). As such, there are strong risks to damage the USRP[2]. See the mailing-list [opee] for several threads on the matter.
- FA-SY2 (`http://www.box73.de/catalog/product_info.php?\-products_id=1870&osCsid=omhrb00t7pga6lqnbq7q3u6ak0`) can be bought for 45 euros, and might theoretically be usable, but nobody has reported using it successfully yet. Check the list.
- Use a clock generator in your lab...

I actually used ***another*** solution: a chip bought by Andy Fung for approximately 13 USD in China. Unfortunately, I do not have the brand nor the precise references for this clock, apart from the specifications listed at Table 1 and the description at Figure 4.

Please note however that this component might not work in your case because, according to Alexander Chemeris:

> *"Usually oscillators with 1ppm staibility is a kind of lottery, because frequency offset may easily become too high for many phones to find your network."*

## 4.3 Installing the clock

Mounting a new 52Mhz clock on an USRP requires a few skills in electronics. If you do not have the proper equipment (soldering station etc) and skills (some components are very small), try and get it done by someone else.

[2]Unless you are using an old USRP board which supports 3.3V.

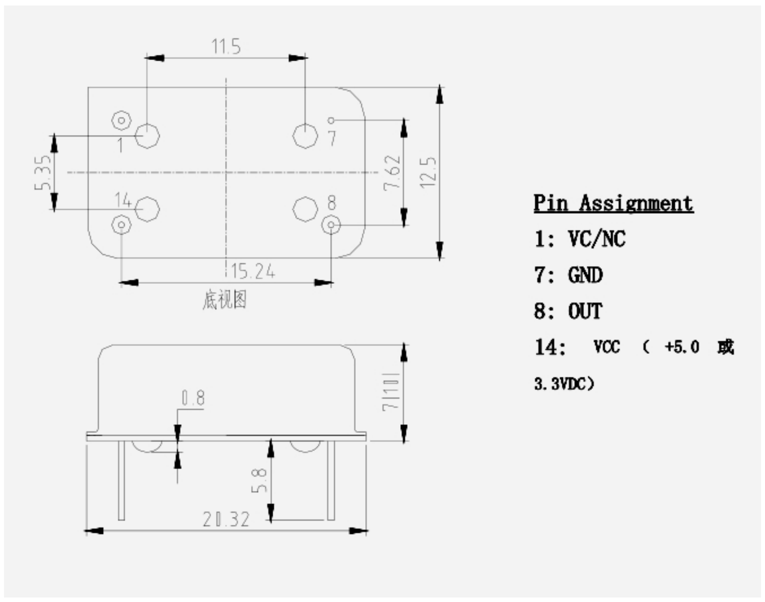


Figure 4: 52 Mhz clock oscillator specifications

The steps to install a new clock are the following. The labels R2039 etc are written on the USRP's board: check where all components are before starting.

- Prepare the USRP to use an external clock [opec]
  - (Difficult step). Move R2029 to R2030T. This disables the onboard clock. R2029/R2030 is a 0-ohm resistor.
  - (Difficult step). Move C925 to C926
  - Remove C924

  NB. I did not need a SMA connector in J2001 nor J2002.
- Connect the new 52Mhz clock: this step is specific to the clock I chose
  - Glue the 52MHz Crystal Oscillator on the USRP mainboard.
  - Solder the red wire (Vcc) and connected it to the 3.3V power source.
  - Solder the black wire (ground) and connected it to the ground point. For example, choose one of the ground pin of J2002.
  - Solder the white wire (52MHz) and connected it the C927.

Note 1. When you solder the wires, make sure the wires are not touching the shell of the crystal (it is a metal case); otherwise, you may create a short circuit.

Note2. The legs of the crystal chip are long. Might need to cut 2/3 out of the legs.

## 4.4 Software patches

Using a 52Mhz requires a few modification of GnuRadio and OpenBTS software and configuration file. I will detail those later (see Section 10.1 and 10.2) but keep this in mind !

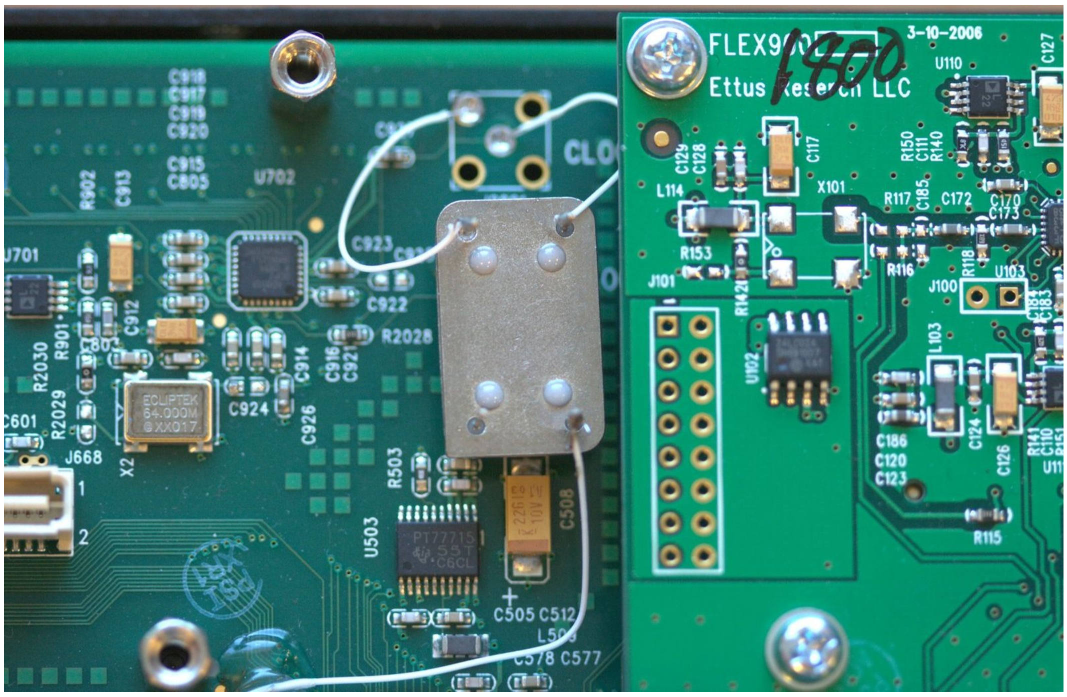


Figure 5: Close shot of the 52 Mhz clock on the USRP

## 5 Software Requirements

The major components you need for OpenBTS are:

- a Linux operating system. It might be portable to other Unix systems (haven't tried). An Ubuntu or a Debian is typically a good choice.
- GnuRadio
- Asterisk
- OpenBTS

### 5.1 Compiling GnuRadio

I use GnuRadio 3.2.2 which is compatible with OpenBTS 2.5.4 Lacassine and OpenBTS 2.6.0 Mamou. But you may wish to build the newer GnuRadio 3.3 (not supported by OpenBTS 2.5.4 Lacassine). The compiling procedure is the same anyway.

1. Install Boost: download from sourceforge

```
./bootstrap.sh --show-libraries
./bootstrap.sh --with-libraries=thread,date_time,program_options
./bjam --prefix=/opt/boost_1_44_0
       builds locally in:
```

| Software | Version |
|---|---:|
| Asterisk | 1.4.21 |
| BOOST | 1.44.0 |
| GnuRadio | 3.2.2 |
| GSL | 1.19 |
| kal | 0.3 |
| libosip2 | 3.3.0 |
| OpenBTS | 2.6.0 Mamou |
| OS | Debian 5.0 Lenny |
| SDCC | 2.9.0 |

Table 2: My own software configuration

```
       /home/work/boost_1_44_0
       /home/work/boost_1_44_0/stage/lib
./bjam --prefix=/opt/boost_1_44_0 install
```

2. Install SDCC from sources. Beware the installation procedure overwrites in `/usr/local/bin` and share, so backup things you might need in there.
3. Install GSL from sources. Do not use the `gsl-bin` package.
4. Install other required packages:

```
apt-get install python-numpy \
python-qt4  libqwt5-qt4-dev qt4-dev-tools \
python-qwt3d-qt4 \
libqwtplot3d-qt4-dev python-qt4-dev \
libxt-dev libaudio-dev libpng-dev \
libxi-dev libxrender-dev libxrandr-dev \
libfreetype6-dev libfontconfig-dev \
python-lxml python-cheetah oss-compat \
swig g++ automake1.9 libtool libusb-dev \
libsdl1.2-dev python-wxgtk2.8 guile-1.8-dev \
libqt4-dev python-opengl fftw3-dev
```

5. Install GnuRadio 3.2.2:
   - download the sources
   - if you are using a 52Mhz clock, there are two patches to apply. The most important one is detailed in section 10.1 or [opec]. Additionally, you should patch the usrp_fft tool (included in GnuRadio) to add a new command line option to set the clock's frequency. See section 10.2.
   - set your library path to:

```
export LD_LIBRARY_PATH=/opt/boost_1_44_0/lib:\
/usr/local/lib:$LD_LIBRARY_PATH
```

- configure specifying to use boost.

  ```
  ./configure --with-boost=/opt/boost_1_44_0
  ```

  It is possible to specify only the components you really need to build. Actually, this is what I tried initially, but, in the end, I think it is a bad idea because you always end up using a component you hadn't expected to and need to recompile it later. Your choice, but you have been warned.

  ```
  ./configure --with-boost=/opt/boost_1_44_0 \
  --disable-all-components --enable-usrp  \
  --enable-omnithread --enable-mblock  \
  --enable-pmt --enable-gnuradio-examples \
  --enable-docs --enable-doxygen \
  --enable-gnuradio-core --enable-gr-wxgui  \
  --enable-gruel --enable-gr-utils \
  --enable-gr-usrp --enable-gr-qtgui
  ```

- make and then make install (if you specified a system directory, make install must be done from root)
- ldconfig

6. Add a USRP group and assign user to that group:

   ```
   addgroup usrp
   addgroup work usrp
   ```

7. Write USRP rules file: /etc/udev/rules.d/10-usrp.rules [Ale09][3]:

   ```
   ACTION=="add", BUS=="usb", SYSFS{idVendor}=="fffe",
   SYSFS{idProduct}=="0002", GROUP:="usrp", MODE:="0660"
   ```

Once you have completed those steps successfully, it is a good idea to test your USRP to see if everything is working as expected (see section 6).

## 5.2 Compiling OpenBTS

To compile OpenBTS 2.6.0 Mamou, follow those steps:

1. download the sources
2. install libosip2-3.3.0 from sources. If you do not intend to use smqueue (SMS), you can install the Debian package libosip2-dev. But if you need smqueue, you'll have to recompile a newer version.
3. install other requirements:

   ```
   apt-get install libortp7-* asterisk
   ```

4. you might have to link boost to local include:

[3] I'm not absolutely certain this is required.

```
ln -s /opt/boost_1_44_0/include/boost /usr/local/include/boost
```

5. set your library path to:

   ```
   export LD_LIBRARY_PATH=/opt/boost_1_44_0/lib: \
   /usr/local/lib:$LD_LIBRARY_PATH
   ```

6. **Patching for old sources only** (this is no longer required for current downloads): if you are using a RFX1800 daughterboard (not RFX 900) you must patch the sources. See section 10.3 (recent snapshots from the git repository do not need this). If you are using a 64Mhz clock (stock), patch Transceiver/USRPDevice files. If you are using 52Mhz clock, patch Transceiver52M/USRPDevice files.

7. **Patching (still required).** if you are using a single daughterboard, you must apply yet another patch. Make sure your daughterboard in on side A. See section 10.4. Again, if you are using a 64Mhz, patch the Transceiver/USRPDevice files. If you are using 52Mhz clock, patch Transceiver52M/USRPDevice files.

8. if you downloaded OpenBTS from sources (repository), run `autoreconf -i` to build the configuration tools.

9. compile OpenBTS (i.e configure and then make)

## 5.3 Compiling smqueue

To have mobile phones in your network send SMS to each other, `smqueue` (included in OpenBTS) must be running. Although smqueue is included in the OpenBTS package, it is not compiled when you build OpenBTS. You must manually invoke smqueue's Makefile:

```
cd ./smqueue
make -f Makefile.standalone
```

Building smqueue requires libosip2 version 3.3.0 or greater. Install it or it will fail to build. Also, you might need g++ v4.3.

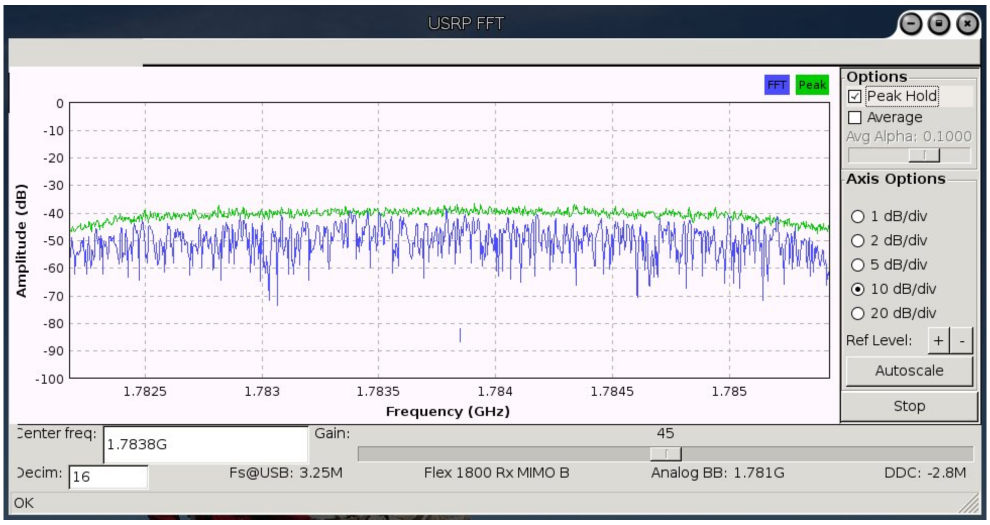
## 6 Testing GnuRadio

### 6.1 USRP Benchmark

Connect the USRP to the computer, compile GnuRadio and then:

```
$ export LD_LIBRARY_PATH=/opt/boost_1_44_0/lib: \
  /usr/local/lib:$LD_LIBRARY_PATH
$ cd /usr/local/share/gnuradio/examples/usrp
$ ./usrp_benchmark_usb.py
```

The script tests USB throughput. You should see several OKs. See [Ale09] for an example.



Center freq: 1.7838G
Gain: 45
Decim: 16
Fs@USB: 3.25M
Flex 1800 Rx MIMO B
Analog BB: 1.781G
DDC: -2.8M
OK

Figure 6: Frequency 1783.8 Mhz is not used

## 6.2 USRP FFT

The usrp_fft tool (in GnuRadio) is useful to test the USRP responds correctly, and also to check whether a given frequency is used or not.

```
$ export LD_LIBRARY_PATH=/opt/boost_1_44_0/lib: \
/usr/local/lib:$LD_LIBRARY_PATH
$ /usr/local/bin/usrp_fft.py
```

Let's see how to check if a given channel is available. First, we need to pick up an available channel for GSM 1800.

[Aub] provides a GSM 1800 uplink and downlink table. For example, if we select channel 880, this correspond to uplink frequency 1783.8 Mhz and downlink frequency 1878.8 Mhz. To use this channel, we need to check those frequencies are not used by someone else.

```
$ usrp_fft.py -f 1.7838G &
```

You should get a relatively "flat" curve such as Figure 6 if the frequency is not used. If it is, pick up another channel.

To test the USRP responds correctly, simulate use of the uplink and downlink frequencies:

```
$ usrp_siggen.py -f 1783.8M
Using TX d'board A: Flex 1800 Tx MIMO B
uU
```

Do not bother about the uU, oOs you may see as output. [Ham08] explains their meaning:

> "a" = audio (sound card) "O" = overrun (PC not keeping up with received data from usrp or audio card) "U" = underrun (PC not providing data quickly enough)
>
> aUaU == audio underrun (not enough samples ready to send to sound card sink) uUuU == USRP underrun (not enough sample ready to send to USRP sink) uOuO == USRP overrun (USRP samples dropped because they weren't read in time.

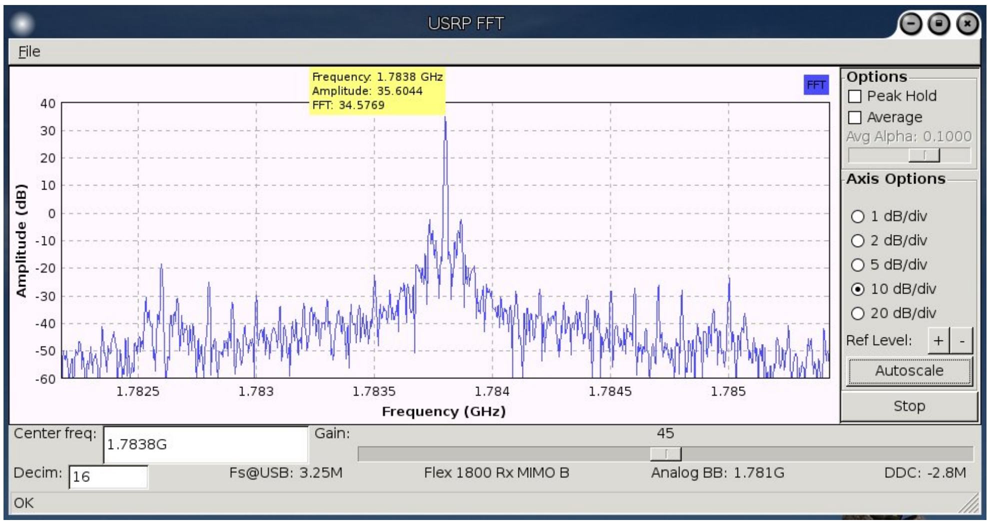


Figure 7: Frequency 1783.8 Mhz is used

Check you see a peak at the corresponding frequency (as in Figure 7).
Do the same for the downlink frequency.

## 6.3 Calibrate the clock

Kal is a tool to calibrate the clock. The latest version (v0.4.1) does not work with GnuRadio 3.2, only GnuRadio 3.3, so you must retrieve an older version from the mailing-list [opea]
To compile kal, install libncurses5-dev:

```
 apt-get install libncurses5-dev
```

Then, run kal.

```
./kal -f 1783800000 -F 52000000 -R A
USRP side: A
FPGA clock: 52000000
Decimation: 192
Antenna: RX2
Sample rate: 270833.343750
error: fcch not detected in 20 frames
```

From Caleb Pal:

*"The fcch error indicate the power level is too low to detect the FCCH and calculate timing offset. Try using an ARFCN of a commercial tower that has a strong(er) signal in your area."*

Finally, note that if the clock source is not adjustable, kal will tell you how much your clock is off by, but you will be unable to do anything about it...

# 7 Configuration

## 7.1 OpenBTS configuration

The OpenBTS configuration file is located in the ./apps directory: openbts.config. See also [Ale09].
Use the default configuration file with the following customization:

- There are two log files (`Log.FileName`) for global logging and for TRX logging. For TRX logging, beware that setting the level to `DEBUG` will cause very heavy logging !

```
Log.Level INFO
Log.FileName openbts26.log
$static Log.FileName
..
TRX.LogLevel INFO
$static TRX.LogLevel
TRX.LogFileName TRX26.log
$static TRX.LogFileName
```

- If you are using a 52Mhz, modify the TRX path to **../Transceiver52M/transceiver**

```
#TRX.Path ../Transceiver/transceiver
TRX.Path ../Transceiver52M/transceiver
$static TRX.Path
```

- Open registration (for a first test)

```
Control.OpenRegistration
$optional Control.OpenRegistration
```

  but then, it probably better not to leave it as such, because this allows any phone to register to your system. Caleb Pal remarks that *"this could introduce legal issues, someone tries to dial the emergency number, can't connect, etc."*

- Set your mobile country code and network code. The country codes are listed at [WMC]. The network codes are listed at [WMN]. Choose the MCC and MNC which corresponds to your SIMs. If you are using a Magic SIM, you can use an used MNC matching your Magic SIM.

```
# 208 = France
GSM.MCC 208
# Unused ? 30
GSM.MNC 30
```

  *To try*: use test MCC/MNC (001/01). Safer than using a real MCC/MNC.

- Set the GSM band and channel. You do not need to set the uplink or downlink frequency.

```
GSM.Band 1800
$static GSM.Band
GSM.ARFCN 880
$static GSM.ARFCN
```

- Notify end-users you do not support emergency calls:

```
GSM.RACH.AC 0x400
...
Control.NormalRegistrationWelcomeMessage Normal Registration Message.
Welcome to OpenBTS! AGPLv3 openbts.sf.net. We do not support
emergency calls. Your IMSI is
```

## 7.2 Get / set your IMSI

SIM cards are identified by their IMSI. If you already have an operational SIM card and want to use it in OpenBTS, you must retrieve its IMSI. To do so, there are several solutions. For instance, use the Python script in [Ale09] or install an application to retrieve it on your mobile phone [opef]. In the latter, beware [opef] wants to send an application registration SMS. Put the mobile offline if you do not want to send that SMS.

If you are using a Magic SIM, you can program it to use a given IMSI using pySIM [Mun]. Plug the USB card writer and SIM to your computer. Set the following:

- argument x for the mobile country code
- argument y for the mobile network code
- argument d for the USB drive the card writer is connected to
- argument t set to 'auto' to automatically detect the card's type

```
./pySim.py -n 26C3 -c 49 -x 208 -y 30
-t auto -z "Random string here" -j 0 -d /dev/ttyUSB0
Generated card parameters :
 > Name    : 26C3
 > SMSP    : 00495...
 > ICCID   : 8949262...
 > MCC/MNC : 208/30
 > IMSI    : 208301111111111111...
 > Ki      : 6ea1ce93440...

Autodetected card type fakemagicsim
Programming ...
Done !
```

## 7.3 Asterisk configuration

See also [Ale09]. You need to set two files: `/etc/asterisk/extensions.conf` and `/etc/asterisk/-sip.conf`.

Basically, at the end of extensions.conf, add one extension per mobile phone you want to add to your network:

```
[sip-local]
exten => 2102,1,Macro(dialSIP,IMSI208123456789012)
exten => 2103,1,Macro(dialSIP,IMSI208555555555555)
```

The numbers 2102 and 2103 are the phone numbers: to call the first mobile phone, dial 2102. To call the second phone number, dial 2103. Of course, you can change the number !

The tag IMSI208123456789012 is like a name for your SIM card. It must match the tag specified in `sip.conf`.

In sip.conf, add one tag per SIM card:

```
[IMSI208123456789012] ; Axelle SIM card IMSI
canreinvite=no
type=friend
context=sip-external
allow=gsm
host=dynamic
```

The tag acts as a section name. You can change it, as long as you use the same name across `sip.conf` and `extensions.conf`.

## 7.4 smqueue

Disable ipv6 for smqueue not to complain about binding to the address.

Configuration of smqueue is located in `./smqueue/smqueue.config`. The default configuration is usually okay except for:

- add `Log.Alarms.Max 10`.otherwise smqueue crashes in some circumstances such as sending a registration SMS.
- create a savedqueue.txt in the `./smqueue` directory.
- run smqueue as **root**. Indeed, smqueue launches asterisk, and tries to read files such as `/etc/asterisk/sip.conf` which are usually only accessible to root.

# 8 Using OpenBTS

## 8.1 Registering phones to the OpenBTS network

If you have completed all previous steps, you can now start using OpenBTS.

1. Plug the OpenBTS box. It's better to put it 40cm away (see Section 9.2) however do not lock it too far away ;)
2. Start OpenBTS and check for any errors.

   ```
   ./apps/OpenBTS
   ```

3. Put your mobile phones on and force the phone to use the OpenBTS network. To do so:

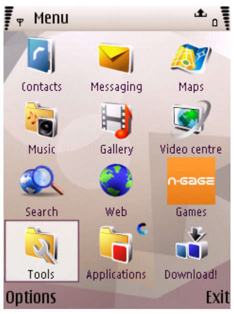


Figure 8: Application menu

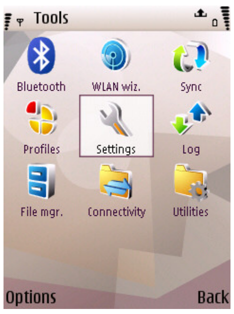


Figure 9: Tools menu

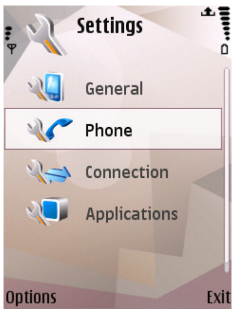


Figure 10: Settings menu

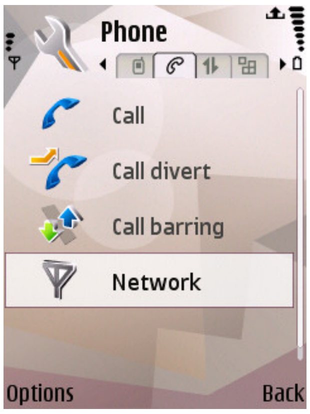
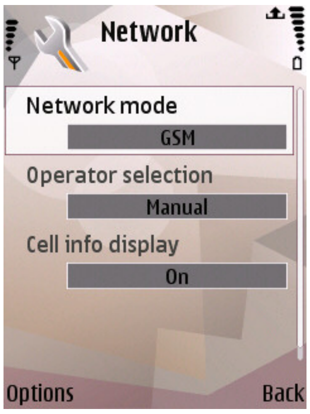


Figure 11: Phone settings



Network mode
GSM
Operator selection
Manual
Cell info display
On

Figure 12: Network settings

- **Nokia**. Go to `tools` (Figure 9), `settings` (Figure 10), then `phone` (Figure 11), then `network` (Figure 12). Select GSM only (not dual mode, nor 3G) and select operator manually (not automatically).
- **iPhone**. Go to `Settings`, then `Operator` and manually scan for available operators.
- **Windows Mobile**. Go to `Settings`, then `Phone`, then `Network`, then select manual scan and select OpenBTS.
- For other phones, see [ope09].

Wait for a while, while the phone is scanning available networks. You should see the OpenBTS network popup. Select it.

4. Wait again, the OpenBTS network should send you a welcome SMS. The exact message depends on the configuration of OpenBTS (openbts.config)

5. You are ready to use the phones. For instance, you can dial 2102 to call another phone (see Figure 15).

## 8.2 Sending SMS

For a quick test, you can send SMS from the OpenBTS console:

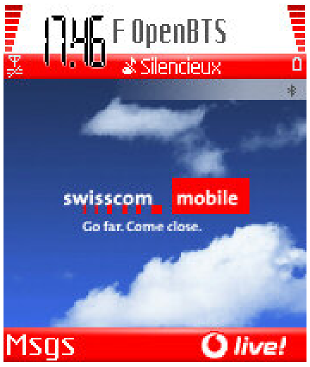
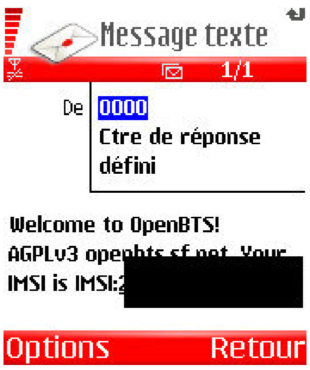


Figure 13: The phone has registered to the OpenBTS network



De 0000
Ctre de réponse défini

Welcome to OpenBTS! AGPLv3 openbts.sf.net. Your IMSI is IMSI:2[redacted]

Figure 14: Welcome message sent to the phone

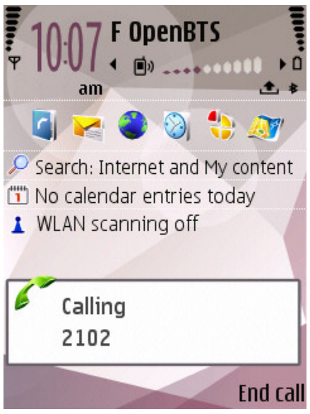


Figure 15: The phone is calling another one

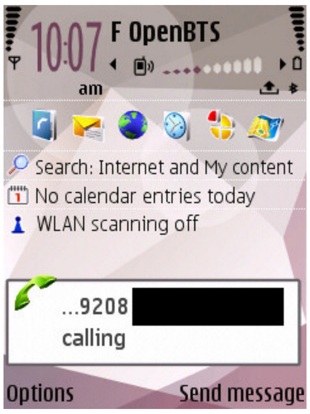


Figure 16: The phone is receiving a call

```
OpenBTS> sendsms 208123456789012 2103
blah blah
```

It is also possible to have phone send SMS to each other:

- **Register the phone**. This step must be done if you get this error in smqueue logs (and the corresponding SMS on your phone):

  ```
  bounce_message: Bouncing 378--vypfd from IMSI208123456789012 to 2103:
  Cannot determine return address; bouncing message.
  Text your phone number to 101 to register and try again.
  ```

  Strangely connecting to the OpenBTS network (step 8.1) does not automatically register the phone, so do as the message says and send an SMS with your own phone number (2102 in my case) to short code 101. You should see the following logs for smqueue (logs are stripped down to keep it short):

  ```
  Got SMS '273--lnobw' from IMSI208123456789012 for 101.
  Responding with "202 Queued".
  Short-code SMS 101(2102).
  answering "Your phone is already registered as  ."
  ```

  The phone you are registering should indeed receive an SMS with this message. Note there is a bug in the message: it does not display the phone number.

- **Send the SMS**. Now that 2102 is registered, the 2102 phone can send an SMS to 2103. When doing so, the smqueue logs should show the following:

  ```
  Got SMS '48--keagw' from IMSI208123456789012 for 2103.
  Responding with "202 Queued".
  75 seconds til Request Destination SIP URL for 48--keagw
  Got SMS 200 Response '48--keagw'.
  Got 200 response for sent msg '48--keagw' in state 7
  Deleting sent message.
  ```

and the SMS should appear on 2103 phone.

## 8.3 TMSIs

Once phones have registered, you should be able to list the Temporary Mobile Subscriber Identities (TMSI):

```
OpenBTS> tmsis
TMSI       IMSI            IMEI               age  used
0x4ce547c0 208123456789012                 ?  113h   84s
0x4ce547c1 208111111111111                 ?  113h  113h

2 TMSIs in table
```

If you want phones to re-register, clear the TMSI table.

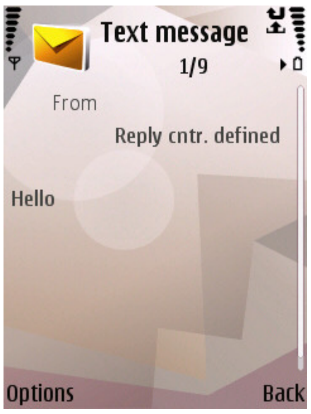

Text message
1/9
From
Reply cntr. defined
Hello

Figure 17: The phone received an SMS

# 9 Miscellaneous

## 9.1 Legal restrictions

Disclaimer: I am absolutely not an expert in this area, please check for yourselves.

Authorities usually regulate the use of some GSM bandwidths. You should check in your country what is authorized, if you need a test license or must use a given channel etc.
In France, the use of GSM bandwidths is regulated by the ARCEP. For limited internal test cases, the use of GSM frequencies is allowed [ARC05]:

> ”Les acteurs non soumis à la déclaration [..] des exploitants de réseaux internes , c’est-à-dire ” entièrement établis sur une même propriété, sans emprunter ni le domaine public y compris hertzien - ni une propriété tierce. ” Ainsi, les réseaux établis par exemple dans les hôtels et les centres commerciaux privés sont exempts de déclaration ;”

Please check for yourselves.

## 9.2 Health

Disclaimer: I am absolutely not an expert in this area, please check for yourselves.

As far as health is concerned, there are two different limits to understand:

- **RF exposure** guildelines are established by the ICNIRP. For 1800Mhz frequencies, they limit exposure to 0.08 W / kg [IEE98]. In practice, the maximum RF exposure levels of a 1 W antenna are already below ICNIRP limits at less than 10cm [Eri04]. OpenBTS uses a 100mW (so even smaller) antenna, so the RF exposure should even be below.
- **Electromagnetic interferences**. The limit is set to 3 V/m (for non-life supporting equipment such as the OpenBTS). In practice, for a 100mW, this limit is met at 40cm or more. This means that further than 40cm there should be only very little risk that the antenna can cause electromagnetic interference with sensitive equipments such as a pacemaker or hearing aids.

Conclusion: having the OpenBTS at 40cm or more of anybody should be really safe?

# 10 Patches

## 10.1 52Mhz patch for GnuRadio

This patch is described at [opec]. If you followed the steps in 5.1, normally, you should have patched your sources. If not, then you absolutely must do it if you are using a 52Mhz clock.

In usrp/host/lib/legacy/usrp_basic.cc, line 116 should read:

```
d_verbose (false), d_fpga_master_clock_freq(52000000), d_db(2)
```

In usrp/host/lib/legacy/usrp_standard.cc, line 1024 should be commented out:

```
// assert (dac_rate() == 128000000);
```

In usrp/host/lib/legacy/db_flexrf.cc, line 179 should read:

```
return 52e6/_refclk_divisor();
```

## 10.2 usrp_fft patch for 52Mhz

This patch adds support for 52Mhz clocks for the usrp_fft GnuRadio tool. Once applied, you may specify the -F option to set the clock's frequency (52000000).

```
diff --git a/gr-utils/src/python/usrp_fft.py b/gr-utils/src/python/usrp_fft.py
index eda9bd5..3bf4ec2 100755
--- a/gr-utils/src/python/usrp_fft.py
+++ b/gr-utils/src/python/usrp_fft.py
@@ -61,6 +61,8 @@ class app_top_block(stdgui2.std_top_block):
                           help="select Rx Antenna (only on RFX-series boards)")
         parser.add_option("-d", "--decim", type="int", default=16,
                           help="set fgpa decimation rate to DECIM [default=%default]")
+        parser.add_option("-F", "--fpga-freq", type="eng_float", default=None,
+                          help="set USRP reference clock frequency to FPGA_FREQ",
+                           metavar="FPGA_FREQ")
         parser.add_option("-f", "--freq", type="eng_float", default=None,
                           help="set frequency to FREQ", metavar="FREQ")
         parser.add_option("-g", "--gain", type="eng_float", default=None,
@@ -99,6 +101,9 @@ class app_top_block(stdgui2.std_top_block):
           #contains 2 Rx paths with halfband filters and 2 tx paths (the default)
           self.u = usrp.source_c(which=options.which, decim_rate=options.decim)

+        if options.fpga_freq is not None:
+            self.u.set_fpga_master_clock_freq(long(options.fpga_freq))
+
         if options.rx_subdev_spec is None:
             options.rx_subdev_spec = pick_subdevice(self.u)
         self.u.set_mux(usrp.determine_rx_mux_value(self.u, options.rx_subdev_spec))
```

## 10.3 GSM 1800 patch

This patch is to use OpenBTS with RFX 1800 daughterboards. It is *not* required:

- for RFX900 daughterboards
- for recent snapshots of OpenBTS (e.g from the git repository, it works straight away)

Note the patch below modifies the USRPDevice files in the Transceiver directory. If you are using a 52Mhz, this directory is not used, it's the Transceiver52M directory you need to patch.

```
--- Transceiver/USRPDevice.cpp 2010-02-25 10:12:28.000000000 +0100
+++ Transceiver/USRPDevice.cpp 2010-07-08 15:01:00.334544272 +0200
@@ -59,7 +59,14 @@
        unsigned *N,
        double *actual_freq)
 {
-
+  if (freq < 1.2e9) {
+    DIV2 = 1;
+    freq_mult = 2;
+  } else {
+    DIV2 = 0;
+    freq_mult =1;
+  }
+
   float phdet_freq = 64.0e6/R_DIV;
   int desired_n = (int) round(freq*freq_mult/phdet_freq);
   *actual_freq = desired_n * phdet_freq/freq_mult;

--- Transceiver/USRPDevice.h 2010-02-25 10:12:28.000000000 +0100
+++ Transceiver/USRPDevice.h 2010-07-08 14:44:45.870543535 +0200
@@ -126,8 +126,8 @@
   static const unsigned CP2 = 7;
   static const unsigned CP1 = 7;
   static const unsigned DIVSEL = 0;
-  static const unsigned DIV2 = 1;
-  static const unsigned freq_mult = 2;
+  unsigned DIV2;
+  unsigned freq_mult;
   static const unsigned CPGAIN = 0;
   static const float    minFreq = 800e6;
   static const float    maxFreq = 1000e6;
```

## 10.4 Single daughterboard patch

If you are using the USRP with a single daughterboard, and not two daughterboards, you need to patch OpenBTS to have all work done on the same daughterboard.

Note the patch below modifies the USRPDevice files in the Transceiver directory. If you are using a 52Mhz, this directory is not used, it's the Transceiver52M directory you need to patch.

```
--- /openbts-2.6.0Mamou/Transceiver52M/USRPDevice.cpp 2010-11-07 02:06:29.643681105 +0100
+++ /singleRFX900_2.6.0Mamou/Transceiver52M/USRPDevice.cpp 2010-11-12 20:11:18.695717220 +0100
@@ -133,17 +133,17 @@
   if (R==0) return false;

   writeLock.lock();
-  m_uRx->_write_spi(0,SPI_ENABLE_RX_B,SPI_FMT_MSB | SPI_FMT_HDR_0,
+  m_uRx->_write_spi(0,SPI_ENABLE_RX_A,SPI_FMT_MSB | SPI_FMT_HDR_0,
      write_it((R & ~0x3) | 1));
-  m_uRx->_write_spi(0,SPI_ENABLE_RX_B,SPI_FMT_MSB | SPI_FMT_HDR_0,
+  m_uRx->_write_spi(0,SPI_ENABLE_RX_A,SPI_FMT_MSB | SPI_FMT_HDR_0,
      write_it((control & ~0x3) | 0));
   usleep(10000);
-  m_uRx->_write_spi(0,SPI_ENABLE_RX_B,SPI_FMT_MSB | SPI_FMT_HDR_0,
+  m_uRx->_write_spi(0,SPI_ENABLE_RX_A,SPI_FMT_MSB | SPI_FMT_HDR_0,
      write_it((N & ~0x3) | 2));
   writeLock.unlock();

-  if (m_uRx->read_io(1) & PLL_LOCK_DETECT)  return true;
-  if (m_uRx->read_io(1) & PLL_LOCK_DETECT)  return true;
+  if (m_uRx->read_io(0) & PLL_LOCK_DETECT)  return true;
+  if (m_uRx->read_io(0) & PLL_LOCK_DETECT)  return true;
   return false;
 }

@@ -249,34 +249,34 @@
   // power up and configure daughterboards
   m_uTx->_write_oe(0,0,0xffff);
   m_uTx->_write_oe(0,(POWER_UP|RX_TXN|ENABLE), 0xffff);
-  m_uTx->write_io(0,(~POWER_UP|RX_TXN),(POWER_UP|RX_TXN|ENABLE));
-  m_uTx->write_io(0,ENABLE,(RX_TXN | ENABLE));
+
+  m_uTx->write_io(0, ENABLE, ENABLE | RX_TXN | POWER_UP); /* power up inverted */
   m_uTx->_write_fpga_reg(FR_ATR_MASK_0 ,0);//RX_TXN|ENABLE);
   m_uTx->_write_fpga_reg(FR_ATR_TXVAL_0,0);//,0 |ENABLE);
   m_uTx->_write_fpga_reg(FR_ATR_RXVAL_0,0);//,RX_TXN|0);
   m_uTx->_write_fpga_reg(40,0);
   m_uTx->_write_fpga_reg(42,0);
-  m_uTx->set_pga(0,m_uTx->pga_max()); // should be 20dB
-  m_uTx->set_pga(1,m_uTx->pga_max());
+  m_uTx->set_pga(0,m_uTx->pga_min()); // should be 20dB
+  m_uTx->set_pga(1,m_uTx->pga_min());
   m_uTx->set_mux(0x00000098);
-  LOG(INFO) << "TX pgas: " << m_uTx->pga(0) << ", " << m_uTx->pga(1);
+
   writeLock.unlock();

   if (!skipRx) {
     writeLock.lock();
-    m_uRx->_write_fpga_reg(FR_ATR_MASK_0  + 3*3,0);
-    m_uRx->_write_fpga_reg(FR_ATR_TXVAL_0 + 3*3,0);
-    m_uRx->_write_fpga_reg(FR_ATR_RXVAL_0 + 3*3,0);
+    m_uRx->_write_fpga_reg(FR_ATR_MASK_0  + 1*3,0);
+    m_uRx->_write_fpga_reg(FR_ATR_TXVAL_0 + 1*3,0);
+    m_uRx->_write_fpga_reg(FR_ATR_RXVAL_0 + 1*3,0);
+    m_uRx->_write_fpga_reg(41,0);
     m_uRx->_write_fpga_reg(43,0);
-    m_uRx->_write_oe(1,(POWER_UP|RX_TXN|ENABLE), 0xffff);
-    m_uRx->write_io(1,(~POWER_UP|RX_TXN|ENABLE),(POWER_UP|RX_TXN|ENABLE));
-    //m_uRx->write_io(1,0,RX2_RX1N); // using Tx/Rx/
-    m_uRx->write_io(1,RX2_RX1N,RX2_RX1N); // using Rx2
-    m_uRx->set_adc_buffer_bypass(2,true);
-    m_uRx->set_adc_buffer_bypass(3,true);
-    m_uRx->set_pga(2,m_uRx->pga_max()); // should be 20dB
-    m_uRx->set_pga(3,m_uRx->pga_max());
-    m_uRx->set_mux(0x00000032);
+    m_uRx->_write_oe(0,0,0xffff);
+     m_uRx->_write_oe(0,(POWER_UP|RX2_RX1N|ENABLE), 0xffff);
+     m_uRx->write_io(0, ENABLE | RX2_RX1N, ENABLE | RX2_RX1N | POWER_UP); /* power up inverted */
+     m_uRx->set_adc_buffer_bypass(0,true);
+     m_uRx->set_adc_buffer_bypass(1,true);
+     m_uRx->set_pga(0,m_uRx->pga_max()); // should be 20dB
+     m_uRx->set_pga(1,m_uRx->pga_max());
+     m_uRx->set_mux(0x00000010);
      writeLock.unlock();
      // FIXME -- This should be configurable.
      setRxGain(47); //maxRxGain());
@@ -312,8 +312,8 @@
   if (!m_uTx) return false;

   // power down
-  m_uTx->write_io(0,(~POWER_UP|RX_TXN),(POWER_UP|RX_TXN|ENABLE));
-  m_uRx->write_io(1,~POWER_UP,(POWER_UP|ENABLE));
+  m_uTx->write_io(0, POWER_UP, (POWER_UP|ENABLE));
+  m_uRx->write_io(0, POWER_UP, (POWER_UP|ENABLE));

   delete[] currData;

@@ -361,8 +361,7 @@
         m_uRx->set_pga(2,0);
         m_uRx->set_pga(3,0);
    }
-   m_uRx->write_aux_dac(1,0,
-        (int) ceil((1.2 + 0.02 - (dB/rfMax))*4096.0/3.3));
+   m_uRx->write_aux_dac(0,0,(int) ceil((1.2 + 0.02 - (dB/rfMax))*4096.0/3.3));

    LOG(DEBUG) << "Setting DAC voltage to " << (1.2+0.02 - (dB/rfMax)) << " "
<< (int) ceil((1.2 + 0.02 - (dB/rfMax))*4096.0/3.3);
```

# 11 Troubleshooting

- **usrp_fft is very slow** In my case, this occurred in two situations: a) when I had forgotten to apply the 52Mhz patch to GnuRadio (see section 10.1) - in that case usrp_fft was really very very slow and b) when I check the button to keep the peaks or average.

  Also, make sure you applied the usrp_fft patch, section 10.2.

- **Impossible to set the frequency to xxxx Mhz !** In my case, this occurred when I had forgotten to apply the 52Mhz patch to GnuRadio (see section 10.1).

- **OpenBTS logs say TX fail to tune** In my case, this occured when I had not applied the GSM 1800 patch (see section 10.3).

- **OpenBTS logs complain about not being able to set the RX or TX gain** In my case, this occurred when the OpenBTS configuration file was wrong and was using the Transceiver directory instead of Transceiver52M (52Mhz clocks).

  ```
  TRX.Path ../Transceiver52M/transceiver
  ```

- **smqueue reports an error "Address already in use"**. Disable ipv6 on the Linux box and the error should disappear.

- **smqueue crashes with the following message:**

  ```
  cannot find configuration value Log.Alarms.Max
  terminate called after throwing an instance of 'ConfigurationTableKeyNotFound
  Aborted
  ```

  The message gives you the answer: add a Log.Alarms.Max entry to smqueue.config (for example, set to 10).

- **smqueue complains: "Failed to read queue from file savedqueue.txt"**. This is a warning (not a fatal error). Smqueue will still run. Create savedqueue.txt in the ./smqueue directory to remove the warning next times.

- **smqueue complains "sh: asterisk: command not found"**. Are you running smqueue as root? Usually, it does not find asterisk because the process is not root and can't access it. Asterisk is typically located in /usr/sbin/asterisk.

- **I've got a 64Mhz clock and my phones are not registering** Please do get a 52Mhz clock, see section 4. My understanding is that it hardly never works with 64 Mhz.

- **Where did you buy your cheap clock?** As I said in section 4, I got it from Andy Fung who got it from a local Chinese retailer, and unfortunately we don't have any better references for it except the specifications copy-pasted in this document. I'm afraid I can't help you find such a clock, but I'm sure there should be others.

- **I've done everything just the same as you but it's not working!** I'm sorry to hear that. Besides getting help from the OpenBTS mailing-list [opee], I would recommend you check:

  1. you have applied all software patches (4)
  2. switch your phones to manual GSM-only network research
  3. the OpenBTS config file uses the Transceiver52M directory (for people using 52Mhz clock)

## 12 Acknowledgements

I definetely need to thank Andy Fung for helping me with OpenBTS. Andy helped step by step understand and build the whole system - not to mention he sent me his spare 52Mhz clock. Yes, Andy, you deserve your bottle of wine with French (smelly) cheese whenever you come and see us in France.

I also wish to thank Alexandre Becoulet for installing the 52Mhz clock on the USRP. This was far beyond my skills and he kindly accepted to do it besides his own work.

I would also thank the OpenBTS mailing-list, in particular Sylvain Munaut, Alexander Chemeris, Caleb Pal, but several other people too who kindly answered my inquiries and keep the mailing-list active.

Finally, thanks to David Maciejak for the nice macro pictures of the USRP, SIM card and reader :)

## References

[Ale09] Alexsander Loula. OpenBTS: Installation and Configuration Guide, May 2009. v0.1.

[ARC05] ARCEP. Les acteurs non soumis à la déclaration, 2005. `http://www.arcep.fr/index.php?id=8055\#c7795`.

[Aub] Aubraux. Arfcn calculator. `http://www.aubraux.com/design/arfcn-calculator.php`.

[Eri04] Ericsson. Radio Waves and Health - In building solutions, 2004. `http://www.ericsson.com/ericsson/corporate_responsibility/health/files/English/EN_brochure_Inbuilding_solutions_2004.pdf`.

[Ham08] Firas Abbas Hamza. The USRP under 1.5X Magnifying Lens!, June 2008. `http://www.scribd.com/doc/9688095/USRP-Documentation`.

[IEE98] IEEE. IEEE Standard for Safety Levels with Respect to Human Exposure to Radio Frequency Electromagnetic Fields, 3 kHz to 300 GHz, 1998. `http://www.icnirp.org/documents/emfgdl.pdf`.

[Mun] Sylvain Munaut. pySIM. `http://git.osmocom.org/gitweb?p=pysim.git;a=summary`.

[opea] Kal v0.3. `http://sourceforge.net/mailarchive/attachment.php?list_name=openbts-discuss&message_id=AANLkTimOp6tMUb69OaCtoEaf5x71ZPVZeYiAryEGCTAK%40mail.gmail.com&counter=1`.

[opeb] OpenBTS. `https://secure.wikimedia.org/wikipedia/en/wiki/OpenBTS`.

[opec] OpenBTS Clock Modifications. `http://gnuradio.org/redmine/wiki/gnuradio/OpenBTSClockModifications`.

[oped] OpenBTS Clocks. `http://sourceforge.net/apps/trac/openbts/wiki/OpenBTS/Clocks`.

[opee] OpenBTS Discuss mailing-list. `http://sourceforge.net/mailarchive/forum.php?forum_name=openbts-discuss`.

[opef] PhoneInfo. `http://www.newlc.com/en/phoneinfo`.

[ope09] FAQ for the Burning Man 2009 Papa Legba Test Network, 2009. `http://sourceforge.net/apps/trac/openbts/wiki/OpenBTS/BM2009FAQ`.

[WMC] List of mobile country codes. `https://secure.wikimedia.org/wikipedia/en/wiki/List_of_mobile_country_codes`.

[WMN] Mobile Network Code. https://secure.wikimedia.org/wikipedia/en/wiki/Mobile_Network_Code.