# OsmoTRX - Feature #2430

## osmo-trx support bladerf

08/10/2017 02:33 PM - Vladimir

| | | | |
|---|---|---|---|
| **Status:** | New | **Start date:** | 08/10/2017 |
| **Priority:** | Low | **Due date:** | |
| **Assignee:** | | **% Done:** | 0% |
| **Category:** | | | |
| **Target version:** | | | |
| **Spec Reference:** | | | |

**Description**

Anyone can add support for a bladerf for osm-trx
I found these files, but when you build different errors come out

### History

**#1 - 08/10/2017 02:34 PM - Vladimir**

Vladimir wrote:

> Anyone can add support for a bladerf for osm-trx
> I found these files, but when build different errors come out

**#2 - 08/15/2017 05:52 PM - laforge**

*- Tracker changed from Support to Feature*

*- Priority changed from Normal to Low*

We'd be more than happy to merge any patches for BladeRF support. However, some of the users/proponents of OsmoTRX on BladeRF would have to develop the related code, test it and submit patches, preferrably via gerrit.osmocom.org.

**#3 - 02/01/2018 08:09 PM - roox**

Here are some notes from my research on this topic:

History of the bladeRF compatible transceiver:

- In **March 2014** @robertghilduta (Robert Ghilduta, Nuand LLC) released the initial transceiver code based on Tranceiver52M
  - https://github.com/Nuand/YateBTS/commit/d0cb6b2863abb5d0ffadd1cd4770b3df52365317 * In **April 2014** @robertghilduta switched to TransceiverRAD1 since he had some problems with Transceiver52M
  - https://github.com/Nuand/YateBTS/commit/36cb7dd06dcf33647ba3307781ff9cf1a271872a * Until **June 2015** the TransceiverRAD1 based transceiver code was used and extended with minor improvements by the yate-people as part of yateBTS
  - http://yate.null.ro/websvn/revision.php?repname=yatebts&path=%2F&rev=503 (svn rev503 is the last version before this kind of transceiver was dropped)
  - *Major changes*:
  - Drop the openBTS-style reading of the configuration parameters via the sqlite-db. Read those parameters from a file instead.
  - Add capability for bladeRF to use USB 2 High Speed mode
  - Add some functions for easier troubleshooting bladeRf-RF-related things * Since late **June 2015** the yate-people are using a new transceiver for yateBTS and the TransceiverRAD1 based transceiver was removed from their repository
  - TransceiverRAD1 delete commit: http://yate.null.ro/websvn/revision.php?repname=yatebts&path=%2F&rev=504
  - New transceiver code: http://yate.null.ro/websvn/listing.php?repname=yatebts&path=%2Ftrunk%2Ftransceiver%2F&opt=dir
  - *Major changes in the new tranceiver code*:
  - Most parts were completely rewritten (by David Burgess?)
  - The transceiver interface was changed (for some unknown reason) - it's not longer compatible with OsmoBTS or OpenBTS :( * In **December 2017** @robertghilduta (Robert Ghilduta, Nuand LLC) released a transceiver that can be used with OpenBTS-UMTS
  - https://github.com/Nuand/OpenBTS-UMTS/commit/7cf94d986c2644b81c5fd900219075d034a68c31
  - It's again based on the code for RAD1-based devices

There are some issues with the bladeRF integration:
The transceiver code thats actually used in osmo-trx evolved from Tranceiver52M (UHD-based devices) while the available tranceiver code for bladeRF is either based on TransceiverRAD1 or uses an incompatible transceiver interface.

I also did some experiments with the last available TransceiverRAD1-based code from:
http://yate.null.ro/websvn/revision.php?repname=yatebts&path=%2F&rev=503

01) Compile yate-bts (rev 503)

```
svn checkout -r 503 http://voip.null.ro/svn/yatebts/trunk yatebts
./configure --enable-bladerf
make
cp ybts.conf.sample mbts/TransceiverRAD1/
```

02) Run the transceiver (it automatically loads FPGA version 0.0.3):
(!) Make sure your bladeRF run with FX3 Firmware version v1.9.1 (v2.0.0 is not compatible!!!)

```
cd yatebts/mbts/TransceiverRAD1/
export MBTSConfigFile=ybts.conf.sample
./transceiver-bladerf
```

03) Run all the other stuff you need for an osmocom-environment
At least: osmo-hlr, osmo-stp, osmo-msc, osmo-mgw, osmo-bsc, osmo-bts

04) **The result**
The transceiver connects to osmo-bts... and RF-stuff will be setup up.
So far so good but...

- Most of the time UEs cannot see the cell
- Even if I have a UE that does see the cell it will not camp on that cell

Here's the from osmo-bsc when a UE tries to connect:

```
<0004> abis_rsl.c:1288 (bts=0,trx=0,ts=0,ss=0) state ACTIVATION REQUESTED -> ACTIVE
<0004> abis_rsl.c:1883 BTS 0 CHAN RQD: reason: answer to paging (ra=0x25, neci=0x00, chreq_reason=0x01)
<0000> chan_alloc.c:412 (bts=0,trx=0,ts=0,pchan=CCCH+SDCCH4) Allocating lchan=1 as SDCCH
<0004> abis_rsl.c:1965 (bts=0,trx=0,ts=0,ss=1) Activating ARFCN(870) SS(1) lctype SDCCH r=PAGING ra=0x25 ta=0
<0004> abis_rsl.c:596 (bts=0,trx=0,ts=0,pchan=CCCH+SDCCH4) Tx RSL Channel Activate with act_type=INITIAL
<0004> abis_rsl.c:625 (bts=0,trx=0,ts=0,ss=1) state NONE -> ACTIVATION REQUESTED
<0004> abis_rsl.c:1629 (bts=0,trx=0,ts=0,ss=1) CHANNEL ACTIVATE ACK
<0004> abis_rsl.c:1288 (bts=0,trx=0,ts=0,ss=1) state ACTIVATION REQUESTED -> ACTIVE
<0004> abis_rsl.c:1753 (bts=0,trx=0,ts=0,ss=0) T3101 expired: no response to IMMEDIATE ASSIGN
<0004> abis_rsl.c:870 (bts=0,trx=0,ts=0,ss=0) RF Channel Release due to error: 1
<0004> abis_rsl.c:780 (bts=0,trx=0,ts=0,ss=0) DEACTivate SACCH CMD
<0004> abis_rsl.c:901 (bts=0,trx=0,ts=0,ss=0) state ACTIVE -> RELEASE DUE ERROR
<0004> abis_rsl.c:942 (bts=0,trx=0,ts=0,ss=0) RF CHANNEL RELEASE ACK
<0004> abis_rsl.c:1753 (bts=0,trx=0,ts=0,ss=1) T3101 expired: no response to IMMEDIATE ASSIGN
<0004> abis_rsl.c:870 (bts=0,trx=0,ts=0,ss=1) RF Channel Release due to error: 1
<0004> abis_rsl.c:780 (bts=0,trx=0,ts=0,ss=1) DEACTivate SACCH CMD
<0004> abis_rsl.c:901 (bts=0,trx=0,ts=0,ss=1) state ACTIVE -> RELEASE DUE ERROR
<0004> abis_rsl.c:942 (bts=0,trx=0,ts=0,ss=1) RF CHANNEL RELEASE ACK
<0004> abis_rsl.c:829 (bts=0,trx=0,ts=0,ss=0) is back in operation.
<0004> abis_rsl.c:830 (bts=0,trx=0,ts=0,ss=0) state RELEASE DUE ERROR -> NONE
<0004> abis_rsl.c:829 (bts=0,trx=0,ts=0,ss=1) is back in operation.
<0004> abis_rsl.c:830 (bts=0,trx=0,ts=0,ss=1) state RELEASE DUE ERROR -> NONE
```

Some years ago others also had no sucess with the original RAD1 devices and osmo-bts and the issues they had were similar to the ones I just described...

- http://lists.osmocom.org/pipermail/openbsc/2014-July/007570.html * https://github.com/kheimerl/vbts-openbts/commits/osmotrx

#### #4 - 02/02/2018 07:50 AM - laforge

Thanks for your research and for the related summary.

I think it could make sense to simply capture a protocol trace on a working yatebts
setup, os one can see the protocol between their "modern" transceiver and yatebts.

This way one could see how the protocol looks like
1) for initialization/control (the CMD + timing mechanism in the old protocol)
2) for actual traffic bursts

Assuming the functional split is still the same (modulation/demodulation in the transceiver,
convolutional coding/decoding in the bts), there would be the option of implementing that
same protocol in OsmoBTS in order to be compatible.

One would then have a configuration option on which protocol dialect to use.

The other approach would be to keep the OsmoTRX code base as-is, but only add the interface on the radio side. This has the advantage of having the same (known) code on all SDR platforms.

So I think I generally see the following three options:

1) implement "new" protocol [as an option] in omso-bts-trx
2) implement libbladerf based support in osmo-trx
3) use soapysdr-module-bladerf, soapysdr and soapy-uhd, similar to how LimeSDR support is currently handled in OsmoTRX

In terms of effort, '3' should be the easiest option. Not elegant, but not really any significant changes to code / architecture / protocols required.

#### #5 - 02/22/2018 09:07 PM - roox

*- File yate-with-yatebts-debug-cli.txt added*

*- File yatebts-modern-tranceiver.pcap added*

Here's some debug output and a pcap trace from a YateBTS session thats using their "modern" transceiver.
The trace includes Radio/BTS setup, MS attach and a MO voice call.

Test environment:

- bladeRF X40 (via USB2)
- Yate v6.0.0
- YateBTS v6.0.0

non-default ybts.conf parameters:

```
Radio.Band=1800
Radio.C0=870
Identity.MCC=001
Identity.MNC=01
Identity.LAC=1
Identity.CI=1
Identity.BSIC.BCC=1
Identity.BSIC.NCC=1
Identity.ShortName=YateBTS
Radio.PowerManager.MaxAttenDB=10
Radio.PowerManager.MinAttenDB=0
...
```

#### #6 - 02/23/2018 03:02 PM - laforge

*- Related to Bug #2975: OsmoBTS doesn't generate measurement indications in absence of uplink bursts added*

#### #7 - 02/23/2018 03:03 PM - laforge

*- Related to deleted (Bug #2975: OsmoBTS doesn't generate measurement indications in absence of uplink bursts)*

#### #8 - 02/23/2018 05:14 PM - laforge

Interesting, thanks for posting. I did a very brief look at the pcap using the new wireshark dissector for "our" old TRX protocol in http://git.osmocom.org/wireshark/log/?h=laforge/trx and it can even still decode some of it.

More research is needed, but it seems that at least fundamentally the concept of ASCII commands in UDP on one port with binary burst data on other UDP ports has remained.

### Files

| | | | |
|---|---|---|---|
| bladeRFDevice.h | 7.48 KB | 08/10/2017 | Vladimir |
| bladeRFDevice.cpp | 34.4 KB | 08/10/2017 | Vladimir |
| configure.ac | 3.7 KB | 08/10/2017 | Vladimir |
| radioInterface.cpp | 9.63 KB | 08/10/2017 | Vladimir |
| radioInterface.h | 5.31 KB | 08/10/2017 | Vladimir |
| yate-with-yatebts-debug-cli.txt | 179 KB | 02/22/2018 | roox |
| yatebts-modern-tranceiver.pcap | 5.7 MB | 02/22/2018 | roox |