TRX SDK

Version: 2018-07-10

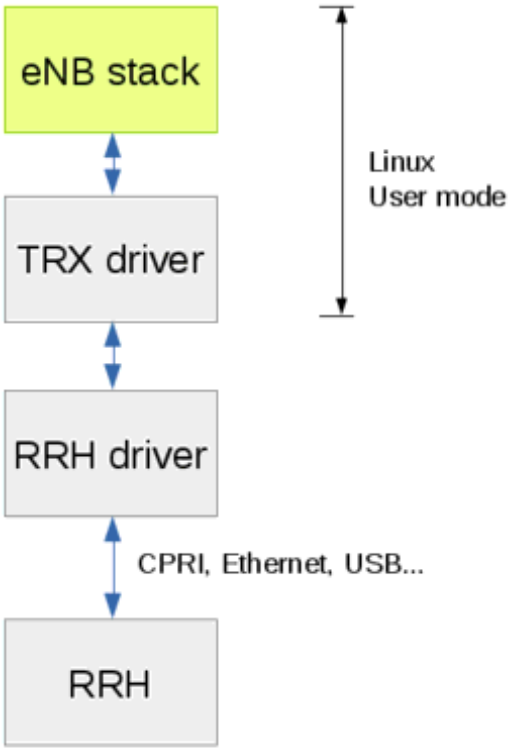
Table of Contents

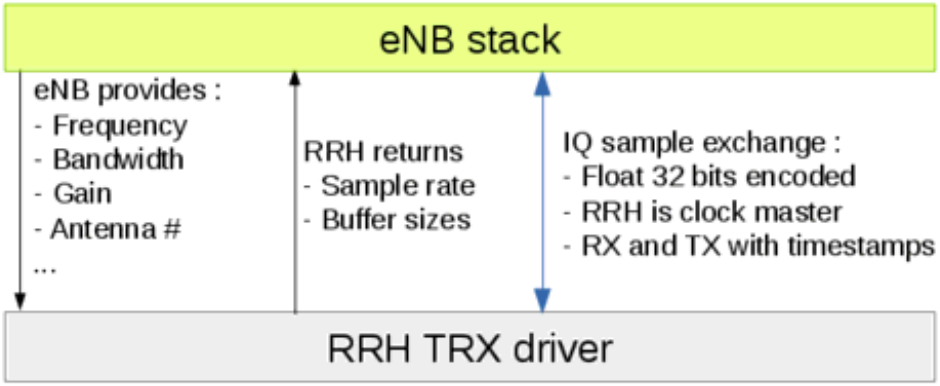
1 Introduction

Here is how to implement a Amarisoft eNB/UE driver to communicate with a radio equipment and exchange IQ samples.
It consists in providing to eNB software a shared library where specific callbacks will be implemented.
You will find an example driver (trx_example.c) to start developing your driver and Makefile to compile it.

This driver may be implemented in any language. It must respect the standard C API provided as `trx_driver.h`

The driver will be launched in user mode.





2 IQ samples

Amarisoft IQ are coded with 32 bits float type and have following range:

```
-1.0 <= IQ <= 1.0
```

It is up to you to adapt level to your specific needs.
As it may require a lot of CPU time, SSE optimized sample code is provided (See [Optimizations], page 6) to help you.

Gain of uplink IQ is direclty handled by eNB software so no specific treatment is needed.

3 Callbacks

Two main callbacks are set to exchange IQ: one for reading and another one for writing.

3.1 RX

Reading process is clock master:

- The read callback (`trx_read_func`) is called regularly to get IQ samples.
- It has to block until at least 1 sample is available and must return at most `count` samples.
- The `count` argument value depends on sample rate and is at most 4096 samples.
- The callback provides clock information using "ptimestamp" pointer.
  It must be expressed in samples so that time can be retrieved by stack using sample rate.
  The timestamp wil be used by stack to provide send time to write process.

3.2 TX

- The write callback (`trx_write_func`) is called regularly and timestamp information is direclty estimated from reading process.
- This timestamp information represent when the radio head must send samples to the air. It is important to be as precise as possible to keep synchronization between TX and RX.
- The `count` argument that represent number of samples to send can be configured by implementing with `trx_get_tx_samples_per_packet_func` callback.
  Else, the stack will estimate this value depending on sample rate
- If `samples` pointer is NULL, this means TX has to be disabled (must not transmit anything) during the time represented by `count` parameter. The flag `TRX_WRITE_FLAG_PADDING` will be also set. This case only happens in TDD mode.
- The write callback must be as fast as possible.

4 Timestamps

Timestamp unit is sample, and is used to convert to time depending on sample rate provided:

```
<time> = <timestamp> / <sample_rate>
```

5 API

Amarisoft driver API is located in trx_driver.h.

6 Example

A dummy driver example is implemented in `trx_example.c`.

`trx_example` is a dummy transceiver driver for the Amarisoft LTE eNodeB. It simulates zero samples coming from a source synchronized to the PC clock and optionally outputs the maximum amplitude of the downlink I/Q samples.

You can compile the trx_example driver by just typing `make`.
Then copy `trx_example.so` to the lteenb directory.
You can enable it with the following property in the eNodeB configuration:

```
//include "rf_driver/1chan.cfg",

rf_driver: {
   name: "example",
   dump_max: 1, /* enable maximum amplitude output */
},
tx_gain: 0,
rx_gain: 0,
sample_rate: 11.52, /* set the sample rate to 11.52 MHz */
```

7 Optimizations

For fast IQ samples convertion, optimized routine are available in `convert16_see.c`.

This is a example code of fixed point/floating point numbers optimized with SSE instructions. This file must be compiled with "-msse4.1" gcc option

For more information on Intel SSE/AVX: `https://software.intel.com/sites/landingpage/IntrinsicsGuide/`