TRX split 7.2 driver example

Version: 2024-12-23

Table of Contents

1 Introduction 1

2 Features 2

3 Network configuration 3
- 3.1 Time synchronization 3
  - 3.1.1 PTP 3
- 3.2 Ethernet optimization 4
- 3.3 VLAN 4

4 Software configuration 5
- 4.1 Linux 5
- 4.2 eNB 5
- 4.3 UE 5
- 4.4 rf_driver 5

5 Troubleshoot 7
- 5.1 Statistics 7
  - 5.1.1 eNB 7
- 5.2 Packet jitter 8

6 trx_s72.c 10
- 6.1 eCPRI transport 10
  - 6.1.1 UDP mode 10
  - 6.1.2 Ethernet mode 10
- 6.2 API Implementation 10
  - 6.2.1 Callbacks 10
    - 6.2.1.1 trx_get_packet_config 10
    - 6.2.1.2 trx_read_packet 11
    - 6.2.1.3 trx_write_packet 11
  - 6.2.2 trx_get_packet 11
  - 6.2.3 Miscellaneous 12

7 Change history 13
- 7.1 Version 2024-09-13 13
- 7.2 Version 2024-06-14 13
- 7.3 Version 2024-03-15 13

8 License 14

1 Introduction

The purpose of this document is to explain how to implement a driver for Amarisoft softwares to use ORAN split 7.2 compatible radio units. Such a driver must implement the TRX API defined in `trx_driver.h` and will be used by Amarisoft software as a shared library.
It is stron recommended to be familiar with the legacy TRX API as documentated here (`https://tech-academy.amarisoft.com/trx_example.doc`).

2 Features

- ORAN category A
- Big Endian byte order
- eAxC
- DU_port: 4 bits
- other: any bits
- Max MTU: 9000
- eCPRI concatenation
- I/Q sample format: raw (16 bits) or block floating point (8, 9, 12 or 14 bits)
- static compression: no compression header supported

3 Network configuration

The driver has been tested with an Intel X710 network interface.
In case of jumbo frame usage, configure your NIC MTU size accordingly.

3.1 Time synchronization

As the system clock is used (CLOCK_REALTIME), the system must be synchronized with the level of accuracy required by the radio unit.

3.1.1 PTP

Here we explain how to configure system as a PTP grandmaster and synchronize system clock to it. This description is not detailed and we recommend to use Internet resources to get more details.

We assume system as at least one network interface supporting PTP.
We assume the radio unit will be synchronize to the system (or same PTP master clock).
We note this interface `if-s72`.

To check `if-s72` PTP support:

```
ethtool -T if-s72
```

To use PTP, install `linuxptp` package (Name may differ depending on Linux distribution).

To configure system as PTP grandmaster, set priority1 and priority2 to 127 (or less) in the global section of your ptp configuration file (/etc/ptp4l.conf) and add a section to use `if-s72`.

Ex:

```
[global]
priority1               127
priority2               127
domainNumber            24
...

[if-s72]
network_transport       L2
hybrid_e2e              0
```

To configure system as PTP client, select the proper domain number set here to 24.
Change `if-s72` by the name of your network interface.

To increase the periodicty of sent ptp announce and sync messages you may set `logAnnounceInterval` and `logSyncInterval` Ex:

```
logAnnounceInterval     -3
logSyncInterval         -4
```

Then launch `ptp4l`. Ex:

```
ptp4l -m -f /etc/ptp4l.conf  -l 7
```

And use `phc2sys` to synchronize clock with interface PTP informations:

```
phc2sys -a -rr -n 24
```

Replace 24 by your domain number

You may disable NTP service to avoid conflict:

```
systemctl disable chronyd
```

```
systemctl stop chronyd
```

You may use your distribution service.
Ex with Fedora: First edit `/etc/sysconfig/phc2sys` to use `-a -rr` command line options and enable/start services:

```
systemctl enable ptp4l
systemctl start ptp4l
systemctl enable phc2sys
systemctl start phc2sys
```

To check phc2sys is working properly (Depends on your Linux distribution), type:

```
journalctl -f -u phc2sys
```

You may see such lines:

```
Jul 17 10:53:55 hostname phc2sys[360200]: [3608400.928] if-s72 sys offset        -9 s2 freq
Jul 17 10:53:56 hostname phc2sys[360200]: [3608401.928] if-s72 sys offset        -8 s2 freq
Jul 17 10:53:57 hostname phc2sys[360200]: [3608402.928] if-s72 sys offset        -7 s2 freq
```

3.2 Ethernet optimization

We recommend to increase MTU of your network interface to have biggest packet as possible.

We also recommend to turn off adaptive TX feature of your network interface:

```
ethtool -C <ifname> adaptive-tx off
```

3.3 VLAN

To use VLANs, you can rely on Linux, it will be transparent.

Ex:

```
ip link add link if-s72 name if-s72.42 type vlan id 42
ifconfig if-s72.42 up
```

This will create a VLAN with id 42 ont `if-s72` interface. To use it just set `bind_interface` to `if-s72.42`

4 Software configuration

4.1 Linux

To send packet on time, as there is no standard mechanism in Linux and hardware to do it, the driverwill poll packets.
For that, Linux sched_rt_runtime_us kernel configuration must be set to -1 (disabled).

```
sysctl -w sched_rt_runtime_us=-1
```

Note that if you have used the Amarisoft OTS installation script and installed OTS service, this will be automatically done.

4.2 eNB

To enable s72 mode, add a s72 section in your rf_port configuration (cf config.enb/config.cfg file).
If you ave installed Amarisoft `lteenb` with automatic installation, check config/s72/config.cfg.

4.3 UE

To enable s72 mode, add a s72 section in your cell configuration (cf config.ue/config.cfg file).
The `t1a_min_cp_dl`, `t1a_max_cp_dl`, `t1a_min_up_dl` and `t1a_min_up_dl` must not be set.

If you ave installed Amarisoft `lteue` with automatic installation, check config/s72/config.cfg.

4.4 rf_driver

`local_sof`
: Optional integer (default = 1). If set to 1, the driver will generate eCPRI private packets, else, those packets are expected to be received from the network.

`clock_factor`
: Optional number (default = 1). If `local_sof` is set, a factor will be applied to the system clock time to allow slowing down (< 1) or speeding up (> 1) the time. Usefull for debug but requires radio unit to support it.

`tai_offset`
: Optional integer (default = computed by Linux). Set the offset in seconds between UTC time and TAI time.

`alpha_offset`
: Optionel integer (default = 0). System Frame Number Calculation alpha offset.

`beta_offset`
: Optionel integer (default = 0). System Frame Number Calculation beta offset.

`rf_ports` Array of object. Each object represent a cell configuration with following parameters:

  `transport`
  : Optional string (default = udp). Can be `udp` or `ethernet`.

  `subcarrier_spacing`
  : Optional integer (default = 30). Cell subcarrier spacing in kHz.

  `max_pdu_size`
  : Optional integer (default = 1472). Maximum eCPRI PDU size in bytes.

`bind_addr`
: String. If `udp` mode is set, defines the IP address to use to send and receive packets.

`bind_interface`
: String. If `ethernet` mode is set, defines the network interface name to use to send and receive packets.

`remote_addr`
: String. If `ethernet` mode is set, defines the destination mac address of the TX ethernet fames.
If `udp` mode is set, defines the destination IP address to send UDP packet to.

`t1a_min_cp_dl`
: Optional integer. ORAN `t1a_min_cp_dl` management parameter in microseconds. If not set, the eCPRI packets will be sent as soon as they are generated.

`t1a_max_cp_dl`
: Optional integer. ORAN `t1a_max_cp_dl` management parameter in microseconds.

`t1a_min_up_dl`
: Optional integer. ORAN `t1a_min_up_dl` management parameter in microseconds.

`t1a_max_up_dl`
: Optional integer. ORAN `t1a_max_up_dl` management parameter in microseconds.

`ta3_max`
: Optional integer. Set time margin in microseconds for incoming U-Plane packet to arrive.
In other words, IQ packets for slot `N` may arrive up to `ta3_max` after start of slot `N+1`

5 Troubleshoot

5.1 Statistics

To check eCPRI/ORAN traffic or to troubleshoot the system, you can use the `t s72` monitor command of `LTEENB` and `LTEUE`.

5.1.1 eNB

On eNB, you will get the following columns prompted:

```
     --User---------------------------------------------- - Slots - ----
Cell total type1 type3   err   inv  miss  late early  lost skip past  err
   1     0     0     0     0     0     1     0     0     0    0    0    0
   1     0     0     0     0     0     1     0     0     0    0    0    0
```

Here is the definition of each column

`Cell` Internal cell ID

`User` User plane statistics:

Info	Description
total	Total number of valid eCPRI packets
type1	Number of valid ORAN section type 1
type3	Number of valid ORAN section type 3
error	Number of unwanted ORAN sections (type 3 only)
inv	Number of ORAN invalid section, i.e wrong header values
miss	Number of non complete OFDM symbols

late	Number of outdated eCPRI packets
early	Number of eCPRI packets arriving too early
lost	Number of eCPRI packets without associated control

`Slots` eCPRI Start Of Frame private packets statistics:

Info	**Description**
skip	Number of dsi-continuities in frame number past
Number packets with frame number in the past	

5.2 Packet jitter

If you have a hard time to get packet being sent in the proper time window and you want to easily get information avout your the global system, we provide a tool inside `OTS` package of Amarisoft software release.
If you have used Amarisoft installation script, it may be located in `/root/ots`. The tool is called lte_toolbox and has a special mode to listen to a network interface and capture `ORAN/eCPRI` packets and estimate their time arrival towards local time.

Ex:

```
./lte_toolbox s72_time eth0
LTE toolbox version 2024-09-13, Copyright (C) 2012-2024 Amarisoft
```

```
Frame number: 1638
--Bitrate--  ----------U-Plane-------------  --------DL C-Plane------------  --------UL C-Pl
               count min   avg max           count min   avg max           count min   avg m
  0.000gbps        0   0   0.0   0  0.0   0       0   0   0.0   0  0.0   0       0   0   0
  0.846gbps    20647 -95 191.0 234 43.8 329    1619 -18 187.9 241 78.0 259     101 -103 -8
  0.836gbps    20400 -23 190.9 234 43.6 257    1600 -17 187.1 241 77.9 258     100 -104 -8
```

6 trx_s72.c

Here are some details about the example code.
For better understading we recommend to also look at the code to get more details and precision.

6.1 eCPRI transport

Two transport mode are supported: `UDP` and `Ethernet`

6.1.1 UDP mode

In this mode, eCPRI PDU are sent inside UDP payloads.
This mode is for testing and has low performances.

6.1.2 Ethernet mode

This mode conforms to ORAN specification by sending eCPRI data inside Ethernet payloads using the `0xaefe` ethernet type.

This mode uses Linux raw socket in PACKET_MMAP (`https://docs.kernel.org/networking/packet_mmap.html`) mode for high performances.

The Ethernet frames may be sent as fast as possible or may be buffered to be sent on time. When sending frames in best effort mode, no time synchronization is needed with the radio unit as the driver expect the radio unit to keep all packets, even if received several slots earlier. To send frames on time, clock synchronization must be ensured.

6.2 API Implementation

When split 7.2 mode is enabled, `trx_start_func2` will be called with `s72_enable` set to true in `TRXDriverParams2`.

6.2.1 Callbacks

Compared to legacy TRX driver, the following methods are mandatory:

- trx_read_packet
- trx_write_packet

The trx_get_packet_config is optional.

The following methods do not apply to split 7.2 implementation:

- trx_read_func, trx_read_func2, trx_read_mt_func
- trx_write_func, trx_write_func2, trx_write_mt_func
- trx_read_timestamp_func
- trx_get_tx_samples_per_packet_func

6.2.1.1 trx_get_packet_config

This method is called after init and before start to provide following informations:

`max_pdu_size`
: Maximum size in byte(s) of eCPRI pdu. Will bound the size of generated packets for TX. If set to 0, the default value is 1472.

`max_read_packets`
: Tells application how maximum number of packets that can be read during `trx_read_packet` call. Default is 1.

`rx_async_release`
: If set to 1, RX packet provided by driver during trx_read_packet call may not be released by application on next call. In this case, trx_read_packet must explicitely check `TRX_READ_PACKET_MD_FLAG_RELEASE` in metadata to be aware of released packets.
Using this option will help to reduce internal jitter/latency and help sending packets on time.

If `trx_get_packet_config` is not defined, default values apply.

6.2.1.2 trx_read_packet

This method is constantly called by application. It may be blocking, waiting for EPCRI packets to be available.
It is up to the driver to allocate packets and provide them using `packets[n].data`.
Up to `count` packets may be provided. Packets provided during a call to this method are considered as non used anymore on next call to this method and thus can be released.
Application is expected to get EPCRI header at `packets(n].data` position.
Length of packet must be set in `packets[n].len`.
eCPRI packets must not be fragmented.
eCPRI message concatenation is supported.
eCPRI packets should be provided as fast as possible.

A private eCPRI packet is expected at the beginning of each new uplink slot, cf build_ecpri_start_of_slot in trx_s72.c to provide application its clock.

`TRXReadPacketMetadata` may be filled this way:

- If `TRX_READ_PACKET_MD_FLAG_OVERFLOW` is set and packets have been lost since last call, set it to 1.

If `rx_async_release` has been set during `trx_get_packet_config`, `packets` pointer will reference `TRXReadPacketMetadata.release_count` packets that can be released.

6.2.1.3 trx_write_packet

This method is called by application to provide generated eCPRI packets to the radio unit.
The eCPRI packets are provided as fast as possible and it is up to the driver to send them at the appropriate time. I.e if the radio unit has limited buffer, it is up to the driver to buffer packets and send them later.
The driver can read the packet information to retrieve timing informations (cf trx_s72_write_packet) to send them at a precise time.

Packets length will be limited by the `max_pdu_size` configuration parameter provided by `trx_get_packet_config`.

`TRXWritePacketMetadat` may be filled this way:

- If `TRX_WRITE_PACKET_MD_FLAG_TIME_BUDGET` is set, indicate by setting `time_budget_set` to 1 how much time remains to send packet to the radio unit in `time_budget_us`.
- If `TRX_WRITE_PACKET_MD_UNDERFLOWS`, set `underflows` to how many packets have been lost.

6.2.2 trx_get_packet

This method is optional. If set, the application will call it to get a buffer to fill eCPRI packet.
`TRXPacketVec.data` must be filled with a buffer of at lease `size` byte(s).
`TRXPacketVec.user_data` may be used to identify packet.

Each generated packet will be passed through trx_write_packet and can be identified by `TRXPacketVec.user_data`. The application won't use the buffer anymore after the call so it is up to the driver to release the buffer.

6.2.3 Miscellaneous

- The gNB/UE applications support discontinuities in packet reception

7 Change history

7.1 Version 2024-09-13

- Added support for TAI offset
- Added multi cell support
- Added `t1a_min_cp_dl`, `t1a_max_cp_dl`, `t1a_min_up_dl` and `t1a_min_up_dl` support
- Added `ta3_max`
- Added `gen_prb0` parameter

7.2 Version 2024-06-14

- Added support for UE simulator
- Added TX packet sending window
- Added async TX for UDP mode

7.3 Version 2024-03-15

- First version

8 License