OCUDU Integrations

Generated 2026-09-05 from 58fa6ca

Integrations

This section covers integrations between OCUDU and third-party components commonly used in 5G network deployments.

ORAN Radio Units

6 items

5G Cores

1 item

Switches and Timing

4 items

Benetel RAN550/RAN650

> **WARNING**
>
> This document is intended to be used as a guide. Variances in firmware and software versions in local setups may require the sample configuration files provided to be changed. As a result please closely follow the specific users guides of your RU in conjunction with this guide.

Overview

This guide provides further details on connecting the OCUDU CU/DU to an RU using the OFH Lib. Specifically the RAN550 and RAN650 RU from Benetel. Those are Split 7.2a indoor and outdoor RUs, respectivly.

> **INFO**
>
> Please refer to the Benetel User Guide Documentation for up-to-date configuration and usage guidelines, along with disclaimer and warranty information. Contact Benetel (sales@benetel.com) for more information.

Configuration

CU/DU

> **INFO**
>
> In earlier versions we've provided config examples for long and short PRACH format. This is not longer required and all default configs now use the short PRACH format.

Sample configuration files for the DU can be found in the `configs` folder of the OCUDU source files.

These configurations are **specific** to the firmware version of the RU. For this guide we've used:

- RAN550_1_v1.4.1-NM-25fa970 for the RAN550
- RAN650_1_v1.4.0-NM-5561771 for the RAN650

The following excerpt shows how the DU is configured to communicate with the RU (for a 4x2 MIMO configuration):

```
ru_ofh:
t1a_max_cp_dl: 470
t1a_min_cp_dl: 419
t1a_max_cp_ul: 336
t1a_min_cp_ul: 285
t1a_max_up: 345
t1a_min_up: 294
ta4_max: 200
ta4_min: 0
is_prach_cp_enabled: true            # Configures if Control-Plane messages should be used
to receive PRACH messages.
compr_method_ul: bfp                 # Uplink compression method.
compr_bitwidth_ul: 9                 # Uplink IQ samples bitwidth after compression.
compr_method_dl: bfp                 # Downlink compression method.
compr_bitwidth_dl: 9                 # Downlink IQ samples bitwidth after compression.
compr_method_prach: bfp              # PRACH compression method.
compr_bitwidth_prach: 9              # PRACH IQ samples bitwidth after compression.
enable_ul_static_compr_hdr: false    # Configures if the compression header is present for
uplink User-Plane messages (false) or not present (true).
enable_dl_static_compr_hdr: false    # Configures if the compression header is present for
downlink User-Plane messages (false) or not present (true).
iq_scaling: 20                       # IQ samples scaling factor applied before compression.
cells:
- network_interface: enp1s0f0      # Ethernet interface name used to communicate with the
RU.
ru_mac_addr: 70:b3:d5:e1:5b:06   # RU MAC address.
du_mac_addr: 80:61:5f:0d:df:aa   # DU MAC address.
vlan_tag_cp: 5                   # VLAN tag value for CP.
vlan_tag_up: 5                   # VLAN tag value for UP.
prach_port_id: [4, 5]            # PRACH eAxC port value.
dl_port_id: [0, 1, 2, 3]         # Downlink eAxC port values.
ul_port_id: [0, 1]               # Uplink eAxC port values.
cell_cfg:
prach:
prach_root_sequence_index: 1
prach_config_index: 159
zero_correlation_zone: 0
ssb:
ssb_block_power_dbm: 0 # 0 for RAN650 with 37dBm Tx power, -11 for RAN550 with 24dBm Tx
power
```

To expand on this, the following parameters are set in the `cells` field:

- `network_interface` : Network interface used to send the OFH packets.
- `ru_mac_addr` : MAC address of the RAN550/650.
- `du_mac_addr` : MAC address of the interface used by the gNB (it should be connected directly to the RU or through a PTP-capable switch).
- `vlan_tag_up/vlan_tag_cp` : V-LAN identifier, should be set to the value configured in the switch settings

RU

Refer to the Benetel User Guide documentation to apply the following configuration changes so that they match your local setup. The following information is purely a guide and may not work for your

specific set up.

We currently recommend to use the following RU configuration as other parameter combinations are untested and seem to cause issues with Benetel RUs:

- 4x2 MIMO configuration with 100 MHz bandwitdh

Ensure the RU is running before trying to make any configuration changes. Some of the values can be set/modified in the /etc/ru_config.cfg file, but some others need to be modified

- `MAC Address` : The MAC address of the DU must be configured in the RU for Control-Plane and User-Plane traffic. In our configuration we use the same MAC address for both planes.
- `VLAN tag` : In our setup the same VLAN ID is used for all network traffic, as only one MAC address is used.
- `Compression` : Configure dynamic compression. We use BFP9 compression for all uplink and downlink channels. Refer to the Benetel User Guide for details on how to configure compression in the RU.
- `Transmission Power` : Depending on your setup, you may need to alter the transmission power of the RU. For example, in a lab setting with the UE in close proximity to the RU, the default power settings may result in UE saturation.
- `PRACH format` : We recommend using short PRACH format.
- `DL scaling` : We use downlink scaling of 0dB and adjust the iq_scaling parameter in the DU.
- `TDD pattern` : The TDD pattern should be set to the 7-2 format (DDDDDDDSUU). Refer to the Benetel guide for other supported patterns.

The full configuration files we used for these set ups can be found below:

- 4x2 MIMO configuration with 100 MHz bandwidth.

Initializing and connecting to the network

Initializing and connecting to the network is done in the same way as outlined in the general 7.2 RU guide.

Initializing the network

The following steps should be taken to initialize the network:

1. Ensure the RX50 is online and that both the PTP process and RU synchronization are running correctly.
2. Run the CU/DU, making sure that the PTP sync between the DU and the Falcon switch is successful as previously outlined.

```
sudo ./gnb -c gnb_ru_ran550_tdd_n78_100mhz_4x2.yml
```

If the DU connects to the RU successfully, you will see the following output:

```
The PRACH detector will not meet the performance requirements with the configuration {Format
0, ZCZ 0, SCS 1.25kHz, Rx ports 1}.

--== OCUDU gNB (commit 2c67c80) ==--

Connecting to AMF on 127.0.0.5:38412
Initializing the Open Fronthaul Interface for sector#0: ul_compr=[BFP,9], dl_compr=[BFP,9],
prach_compr=[BFP,9], prach_cp_enabled=false, downlink_broadcast=false
Cell pci=1, bw=100 MHz, 4T2R, dl_arfcn=650000 (n78), dl_freq=3750.0 MHz,
dl_ssb_arfcn=647328, ul_freq=3750.0 MHz

==== gNodeB started ===
Type <t> to view trace
```

Connecting to the network

You can now connect a UE to the network. This can either be done using e.g. a COTS UE. See the main RU guide for details on this.

An example Amarisoft UE configuration file can be found below:

- UE configuration (2x2 MIMO configuration with 20 MHz bandwidth).

This configuration was tested with a R550 and a **specific** Amarisoft UE version (`lteue-linux-2023-09-08`), whilst using a cabled setup with RF splitters and 30 dB attenuation between the Rx ports of the SDR cards and the R550 antenna ports.

Foxconn RPQN-(48/78)00(E/I)

> **WARNING**
>
> This document is intended to be used as a guide. Variances in firmware and software versions in local setups may require the sample configuration files provided to be changed. As a result please closely follow the specific users guides of your RU in conjunction with this guide.

Overview

This guide provides further details on connecting the OCUDU CU/DU to an RU using the the O-RAN 7.2 split. Specifically, we discus the Foxconn RPQN family of devices. They are available in various flavors, hence different model numbers:

- With and without external antenna (the E letter at the end stands for external antenna and the I for internal).
- For different bands (the 78 means n78 for Europe and 48 means band n48 for US markets).
- There is also the “xx01” series that apperantly has some further differences in RF characterisics.

> **INFO**
>
> Please refer to the Foxconn User Guide provided by the distributor for up-to-date configuration and usage guidelines.

Configuration

CU/DU

A sample configuration file for the DU can be downloaded from here. This configuration is **specific** to the firmware version of the RU (v3_1_13q_551p1) being used for this guide. As a result, you may need to modify this slightly for your local setup.

The following excerpt shows how the DU is configured to communicate with the RU:

```
ru_ofh:
t1a_max_cp_dl: 420
t1a_min_cp_dl: 250
t1a_max_cp_ul: 420
t1a_min_cp_ul: 250
```

```
t1a_max_up: 196
t1a_min_up: 80
ta4_max: 500
ta4_min: 25
is_prach_cp_enabled: true
ignore_ecpri_payload_size: true
compr_method_ul: bfp
compr_bitwidth_ul: 9
compr_method_dl: bfp
compr_bitwidth_dl: 9
compr_method_prach: bfp
compr_bitwidth_prach: 9
enable_ul_static_compr_hdr: false
enable_dl_static_compr_hdr: false
iq_scaling: 1.0
cells:
- network_interface: xxxx          # Ethernet interface name used to communicate with the
RU.
ru_mac_addr: 6c:ad:ad:xx:xx:xx    # RU MAC address.
du_mac_addr: 80:61:5f:xx:xx:xx    # DU MAC address.
vlan_tag_cp: 2                    # VLAN tag value for CP.
vlan_tag_up: 2                    # VLAN tag value for UP.
prach_port_id: [4, 5, 6, 7]       # PRACH eAxC port values.
dl_port_id: [0, 1]                # Downlink eAxC port values.
ul_port_id: [0, 1, 2, 3]          # Uplink eAxC port values.
```

To expand on this, the following parameters are set in the `cells` field:

- `network_interface` : Network interface used to send the OFH packets.
- `ru_mac_addr` : MAC address of the RU.
- `du_mac_addr` : MAC address of the interface used by the DU for OFH traffic.
- `vlan_tag_up/vlan_tag_cp` : VLAN identifier, should be set to the value configured in the RU/switch settings.

RU

Refer to the Foxconn User Guide documentation to apply the following configuration changes so that they match your local setup. The following information is purely a guide and may not work for your specific set up.

The RU can operate and has been tested in bandwidths between 20 to 100 MHz, in SISO mode as well as in 2x2 or 4x4 MIMO mode. In ideal conditions (i.e., in a conducted setup) maximum data rate can be achieved and 256-QAM are supported for both downlink and uplink tranmssions. We've found that the best signal quality is achieved with 90 MHz bandwidth due to some filter roll-off in the RU.

All configuration values of the RU can be set/modified in the `RRHconfig_xran.xml` file. The file can usually be found in the `/home/root/sdcard/` folder but has moved to `/home/root` in more recent firmware versions.

Below are the most critical values to be set and the ones that differ from the RU's default configuration values:

- `RRH_DST_MAC_ADDR` : The MAC address of the DU must be configured in the RU for Control-Plane and User-Plane traffic. Note that the `RRH_SRC_MAC_ADDR` is usually not read from the config file and will be auto-set to the MAC address of the 10G interface during boot.
- `RRH_EN_EAXC_ID` : Set to `0` to map RU port ID 0, 1, 2, 3 to PDSCH/PUSCH and 4, 5, 6, 7 to PRACH.
- `RRH_CMPR_TYPE = 1, 1` : Set to `1, 1` to enable BFP compression for PDSCH/PUSCH and PRACH.
- `RRH_C_PLANE_VLAN_TAG/RRH_U_PLANE_VLAN_TAG` : Set this to a value of your choice, we usually use the same VLAN for both.
- `RRH_MAX_PRB` : The channel bandwidth. Set this to e.g., `51` for a 20 MHz cell, to `245` for 90 MHz and `273` for a 100 MHz cell.
- `RRH_LO_FREQUENCY_KHZ` : The center frequency of the cell in kHz, set to e.g., `3600000, 0` to radiate at 3.6 GHz (ARFCN 640000).
- `RRH_DL_IQ_SCALING` : Only present in newer FW versions (e.g. v3.1.15q.551v0706). Set to `0x10001`. Only tested with 2T2R.

The full configuration example file we used for this set up can be found `here`. In this config we are configuring the RU for 20 MHz.

After the configuration file has been updated the RU firmware can be initialized by running the `./init_rrh_config_enable_cuplane` script on the console. Observe the output, verify that the values have been read from the config file correctly and that the RU locks the PTP sync. For this the init script should print `ptp locked: state=3` at the end of the execution.

Initializing and connecting to the network

Initializing and connecting to the network is done in the same way as outlined in the general 7.2 RU guide.

Initializing the network

The following steps should be taken to initialize the network:

1. Ensure the RU is started and has PTP lock (see above).
2. Run the CU/DU, making sure that the PTP sync between the DU and the Falcon switch is successful as previously outlined.

```
sudo ./gnb -c gnb_ru_rpqn4800e_tdd_n78_20mhz.yml
```

If the DU connects to the RU successfully, you will see the following output:

```
--== OCUDU gNB (commit 2c67c80) ==--

Connecting to AMF on 127.0.0.5:38412
Initializing the Open Fronthaul Interface for sector#0: ul_compr=[BFP,9], dl_compr=[BFP,9],
prach_compr=[BFP,9], prach_cp_enabled=true, downlink_broadcast=false
```

```
Cell pci=1, bw=20 MHz, 2T2R, dl_arfcn=640000 (n78), dl_freq=3600.0 MHz, dl_ssb_arfcn=639648,
ul_freq=3600.0 MHz

==== gNodeB started ===
Type <t> to view trace
```

3. If you have the DU running you can go back to the SSH console on the RU and check that the fronthaul traffic is arriving on time. For this run:

```
tail -f /var/log/rrh_log_print/rrh_log_print.log  | grep xRN
```

The RU should be printing something like the following every second:

```
[2024-05-10 08:28:23.034] xRN: total=2740860 c_early=0 c_on=218300 c_late=0 err_tci=0
err_ecpri=0 err_port=0 err_sct=0 err_total=1443194
```

Verify that the the values in the `c_on` column and the `err_total` column increase but all other columns should be zero. Note the misleading name, in the `err_total` column the RU actually counts for uplane packets as well. This has been fixed in later firmware releases.

Connecting to the network

You can now connect a UE to the network. This can be done using e.g. a COTS UE. See the main RU guide for details on this.

LITEON FlexFi

Overview

This guide provides further details on connecting the OCUDU CU/DU to an RU using the the O-RAN 7.2 split. Specifically, the LITEON FlexFi. We've tested with a model FF-RFI078I4 with firmware version 02.00.09.

Configuration

CU/DU

You can download a sample gNB configuration file that is compatible with the LITEON FlexFI RU here.

This configuration file will allow you to create a 100MHz TDD 2T1R cell in band n78.

RU

The RU needs to be flashed with firmware >= 02.00.09. Older firmware versions will not work correctly.

First, power on and access the RU via the command line. The RU must then be configured and power-cycled before it can be used.

The following commands are used to configure the RU, they are shown with the command first and then the associated output:

```
(config)# compression-bit 9
Old Compression Bit = 8
New Compression Bit = 9

(config)# du-mac-address 80615f0ddfab
Old DU MAC Address = 001122334466
New DU MAC Address = 80615f0ddfab

(config)# jumboframe 1
Old jumboframe = 0x00000000
New jumboframe = 0x00000001

(config)# phasecomp-mode true
phase compensation mode : Enable

(config)# slot-id 1
Old slotid = 0x00000002
New slotid = 0x00000001
```

```
(config)# sync-source PTP
sync source : PTP
Active after reboot

(config)# tick 1
Old tick = 0x00000001
New tick = 0x00000001

(config)# c/u-plane-vlan
control and user Plane vlan = 564

(config)# center-frequency
Center Frequency = 3749700000
```

The RU should now be power-cycled. Once completed successfully, the RU state can be checked with:

```
# show oru-status
Sync State : SYNCHRONIZED
RF State : Ready
DPD : Ready
DuConnected : notReady
```

Note that at this stage the DU is not generating any traffic.

Initializing the network

The following steps should be taken to initialize the network:

1. Ensure the RU is started and has PTP lock (see above).

2. Run the CU/DU, making sure that the PTP sync between the DU and the Falcon switch is successful as previously outlined.

   ```
   sudo ./gnb -c gnb_ru_liteon_tdd_n78_100mhz.yml
   ```

   If the DU connects to the RU successfully, you will see the following output:

   ```
   The PRACH detector will not meet the performance requirements with the configuration {F
   ormat B4, ZCZ 0, SCS 30kHz, Rx ports 1}.

   --== OCUDU gNB (commit 2c67c80) ==--

   Connecting to AMF on 10.12.1.105:38412
   Initializing the Open Fronthaul Interface for sector#0: ul_compr=[BFP,9], dl_compr=
   [BFP,9], prach_compr=[BFP,9], prach_cp_enabled=true, downlink_broadcast=false
   Cell pci=1, bw=100 MHz, 2T1R, dl_arfcn=649980 (n78), dl_freq=3749.7 MHz,
   dl_ssb_arfcn=647232, ul_freq=3749.7 MHz
   ```

```
==== gNodeB started ===
Type <t> to view trace
```

3. If you have the DU running you can go back to the SSH console on the RU and check that the fronthaul traffic is arriving on time. For this run:

```
# show oru-status
Sync State : SYNCHRONIZED
RF State : Ready
DPD : Ready
DuConnected : Ready
```

`DuConnected` is now Ready indicating that the RU is receiving traffic and radiating. The RU performance metrics can be checked with:

```
# show pm-data
1,POWER,2024-06-26T17:09:53Z,2024-06-26T17:10:10Z,o-ran-hardware:O-RU-
FPGA,8.6181,9.5173,8.8625,iana-hardware:cpu,8.6181,9.5173,8.8625
2,TEMPERATURE,2024-06-26T17:09:53Z,2024-06-26T17:10:10Z,o-ran-hardware:O-RU-
FPGA,62.0732,64.5135,63.5156,iana-hardware:cpu,62.5706,64.9642,63.4276
13,VOLTAGE,2024-06-26T17:09:53Z,2024-06-26T17:10:27Z,0,0.0000,2024-06-
26T17:09:54Z,1.8311,2024-06-26T17:09:55Z,0.0000,2024-06-26T17:09:54Z,0.0000,2024-06-
26T17:10:27Z,3749700000
1,POWER,2024-06-26T17:10:10Z,2024-06-26T17:10:27Z,o-ran-hardware:O-RU-
FPGA,8.6181,9.6016,9.0758,iana-hardware:cpu,8.6181,9.6016,9.0758
2,TEMPERATURE,2024-06-26T17:10:10Z,2024-06-26T17:10:27Z,o-ran-hardware:O-RU-
FPGA,62.5085,64.7777,63.5053,iana-hardware:cpu,62.0111,64.9021,63.2120
1,RX_ON_TIME,2024-06-26T17:09:53Z,2024-06-26T17:10:27Z,ru1,6863018
2,RX_EARLY,2024-06-26T17:09:53Z,2024-06-26T17:10:27Z,ru1,0
3,RX_LATE,2024-06-26T17:09:53Z,2024-06-26T17:10:27Z,ru1,0
6,RX_TOTAL,2024-06-26T17:09:53Z,2024-06-26T17:10:27Z,ru1,7354134
7,RX_ON_TIME_C,2024-06-26T17:09:53Z,2024-06-26T17:10:27Z,ru1,490546
8,RX_EARLY_C,2024-06-26T17:09:53Z,2024-06-26T17:10:27Z,ru1,0
9,RX_LATE_C,2024-06-26T17:09:53Z,2024-06-26T17:10:27Z,ru1,0
1,TX_TOTAL,2024-06-26T17:09:53Z,2024-06-26T17:10:27Z,ru1,3976
```

Verify that the the values in the `RX_ON_TIME`, `RX_ON_TIME_C` and `TX_TOTAL` column increase but all other columns should be zero.

Connecting to the network

You can now connect a UE to the network. This can be done using e.g. a COTS UE. See the main RU guide for details on this.

Pegatron O-RU (PR1450/PR2850)

> **WARNING**
>
> This document is intended to be used as a guide. Variances in firmware and software versions in local setups may require the sample configuration files provided to be changed. As a result please closely follow the specific users guides of your RU in conjunction with this guide.

Overview

This guide provides further details on connecting the OCUDU CU/DU to an RU using the O-RAN 7.2 split. Specifically, we discuss the Pegatron family of O-RAN Radio Units. Pegatron 5G offers several RU models covering different deployment scenarios:

- **PR1450-78I**: Indoor 5G NR FR1 4T4R O-RU for private network deployments. (Tested Firmware: v1.0.2.4p1)

> **Note:** These Pegatron RUs are built upon a Metanoia reference design. Therefore, the configurations and troubleshooting steps outlined in this guide will likely apply to other Metanoia-based RUs.

> **INFO**
>
> Please refer to the Pegatron 5G User Guide provided by the distributor for up-to-date configuration and usage guidelines. Contact Pegatron (https://5g.pegatroncorp.com/) for more information.

Configuration

CU/DU

A sample configuration file for the DU can be found in the `configs` folder of the OCUDU source files. This configuration is **specific** to the firmware version of the RU being used for this guide. As a result, you may need to modify this slightly for your local setup.

The following excerpt shows how the DU is configured to communicate with the RU (for a 4x4 MIMO configuration with 100 MHz bandwidth):

```
ru_ofh:
t1a_max_cp_dl: 470
```

```
t1a_min_cp_dl: 285
t1a_max_cp_ul: 429
t1a_min_cp_ul: 285
t1a_max_up: 350
t1a_min_up: 125
ta4_max: 180
ta4_min: 110
is_prach_cp_enabled: true
compr_method_ul: bfp
compr_bitwidth_ul: 9
compr_method_dl: bfp
compr_bitwidth_dl: 9
compr_method_prach: bfp
compr_bitwidth_prach: 9
enable_ul_static_compr_hdr: true
enable_dl_static_compr_hdr: true
ru_reference_level_dBFS: -12
subcarrier_rms_backoff_dB: 0

cells:
- network_interface: 0000:ca:01.3
ru_mac_addr: 48:21:0b:xx:xx:xx
du_mac_addr: 00:11:22:33:44:55
vlan_tag_cp: 5
vlan_tag_up: 5
enable_promiscuous: true
check_link_status: false
prach_port_id: [4, 5, 6, 7]
dl_port_id: [0, 1, 2, 3]
ul_port_id: [0, 1, 2, 3]
```

To expand on this, the following parameters are set in the `cells` field:

- `network_interface` : PCI address of the DPDK-bound network interface used to send the OFH packets. When using SR-IOV, this is the Virtual Function (VF) PCI address.
- `ru_mac_addr` : MAC address of the Pegatron RU.
- `du_mac_addr` : MAC address of the interface used by the DU for OFH traffic.
- `vlan_tag_up/vlan_tag_cp` : VLAN identifier, should be set to the value configured in the RU/switch settings.
- `enable_promiscuous` : Must be set to `true` when using SR-IOV Virtual Functions if the VF MAC does not match the `du_mac_addr`. Alternatively, you can set this to `false` if you configure the VF MAC to match the PF MAC at the OS level.
- `check_link_status` : Set to `false` for SR-IOV VF interfaces where link status reporting may not be supported.

RU

Refer to the Pegatron 5G User Guide documentation to apply the following configuration changes so that they match your local setup. The following information is purely a guide and may not work for your specific set up.

The RU has been tested with the following configuration:

- 4x4 MIMO configuration with 100 MHz bandwidth on band n78

Below are the most critical values to be configured on the RU side, and the ones that should match the DU configuration:

- `DU MAC Address` : The MAC address of the DU must be configured in the RU for Control-Plane and User-Plane traffic. In our configuration we use the same MAC address for both planes.
- `VLAN tag` : In our setup the same VLAN ID (5) is used for all network traffic (C-Plane and U-Plane), as only one MAC address is used.
- `Compression` : Configure **static** BFP9 compression for all uplink and downlink channels. The Pegatron RU uses static compression headers (i.e. the compression header is **not** included in each OFH message), which is why `enable_ul_static_compr_hdr` and `enable_dl_static_compr_hdr` are both set to `true` on the DU side.
- `eAxC Port Mapping` : Configure RU port IDs 0, 1, 2, 3 for PDSCH/PUSCH and 4, 5, 6, 7 for PRACH.
- `Center Frequency` : Set to match the DU `dl_arfcn`. For example, ARFCN 650000 corresponds to 3750.0 MHz.
- `Channel Bandwidth` : Set to 100 MHz (273 PRBs) to match the DU configuration.
- `TDD Pattern` : The TDD pattern should match the DU's `tdd_ul_dl_cfg`. With the configuration above (period 10, 7 DL slots + 6 DL symbols, 2 UL slots + 4 UL symbols), this corresponds to a DDDDDDDSUUU pattern.
- `PTP Synchronization` : Ensure PTP is configured and locked before starting the DU. The RU must have a stable PTP lock for correct fronthaul timing.

DPDK and SR-IOV Configuration

> **INFO**
>
> The specific CPU core isolations, EAL arguments, and SR-IOV configurations shown below are highly dependent on specific server hardware and OS environment. They are provided as a working example rather than a strict requirement for the RU itself.

Unlike the Benetel and Foxconn guides which use kernel-mode Ethernet interfaces, this setup uses **DPDK with SR-IOV**. The `network_interface` field in the DU configuration takes a PCI address (e.g. `0000:ca:01.3`) instead of a Linux interface name (e.g. `enp1xxx`).

To use this mode:

1. Ensure the physical NIC supports SR-IOV and that a Virtual Function (VF) has been created and bound to a DPDK-compatible driver (e.g. `iavf` via `vfio-pci`).
2. Pass the correct DPDK EAL arguments in the `hal` section:

```
hal:
eal_args: "--lcores (0-13)@(3,5,7,11,13,15,17,19,21,23,25,27,29,31) -a 0000:ca:01.3 --file-
prefix=gnb --log-level=lib.eal:error --log-level=pmd:error -d /opt/dpdk/24.11.2/lib/x86_64-
linux-gnu/dpdk/pmds-25.0/librte_common_iavf.so -d /opt/dpdk/24.11.2/lib/x86_64-linux-
```

```
gnu/dpdk/pmds-25.0/librte_net_iavf.so"
```

3. The `-a` flag specifies the PCI address of the VF, and `-d` flags load the required DPDK PMD drivers for Intel Adaptive Virtual Function (iavf).
4. The `--lcores` mapping pins DPDK worker threads to specific CPU cores for real-time performance.

> **INFO**
>
> When deploying on Kubernetes (e.g. StarlingX), the SR-IOV VF is typically allocated via the SR-IOV Device Plugin and the PCI address is injected into the pod at runtime. The `enable_promiscuous: true` and `check_link_status: false` settings are necessary in this scenario.

CPU Affinity and Threading

For optimal real-time performance, the DU should be configured with appropriate CPU affinity:

```
expert_execution:
affinities:
main_pool_pinning: mask
ofh:
timing_cpu: "3"                  # Dedicated CPU core for OFH timing-critical tasks.
threads:
main_pool:
nof_threads: 12                  # Number of worker threads in the main thread pool.
task_queue_size: 2048
backoff_period: 10
```

Ensure that the CPU cores specified are isolated from the OS scheduler (e.g. via `isolcpus` kernel parameter or a CPU manager like the one provided by StarlingX/Wind River).

Initializing and connecting to the network

Initializing and connecting to the network is done in the same way as outlined in the general 7.2 RU guide.

Initializing the network

The following steps should be taken to initialize the network:

1. Ensure the Pegatron RU is powered on and that PTP synchronization is locked.
2. Run the CU/DU, making sure that the PTP sync between the DU and the fronthaul switch is successful as previously outlined.

```
sudo ./gnb -c gnb_ru_pegatron_tdd_n78_100mhz_4x4.yml
```

If the DU connects to the RU successfully, you will see the following output:

```
--== OCUDU gNB (commit xxxxxxx) ==--

Connecting to AMF on 127.0.0.5:38412
Initializing the Open Fronthaul Interface for sector#0: ul_compr=[BFP,9], dl_compr=[BFP,9],
prach_compr=[BFP,9], prach_cp_enabled=true, downlink_broadcast=false
Cell pci=0, bw=100 MHz, 4T4R, dl_arfcn=650000 (n78), dl_freq=3750.0 MHz,
dl_ssb_arfcn=647328, ul_freq=3750.0 MHz

==== gNodeB started ===
Type <t> to view trace
```

Troubleshooting

If the DU fails to connect to the RU, check the following:

- **PTP lock**: Verify that the RU has achieved PTP lock before starting the DU.
- **MAC addresses**: Ensure `ru_mac_addr` and `du_mac_addr` are correct and match the physical setup.
- **VLAN configuration**: Verify that the VLAN tags match between the DU, RU, and any intermediate switches.
- **Compression settings**: The Pegatron RU uses static compression headers. Make sure both `enable_ul_static_compr_hdr` and `enable_dl_static_compr_hdr` are set to `true`.
- **DPDK driver**: Verify the SR-IOV VF is correctly bound and the iavf PMD drivers are loaded.

Connecting to the network

You can now connect a UE to the network. This can be done using e.g. a COTS UE. See the main RU guide for details on this.

Kubernetes / Helm Deployment

This RU has also been tested with the OCUDU Helm chart for Kubernetes-based deployments (e.g. on StarlingX). Key Helm values for this setup include:

- SR-IOV Device Plugin resource requests (`intel.com/pci_sriov_net_fh_cp`)
- Isolated CPU allocation (`windriver.com/isolcpus`)
- Hugepages (1Gi pages recommended for DPDK)
- Multus CNI for N2/N3 user plane connectivity
- Host network disabled (using SR-IOV VFs instead)

Refer to the OCUDU Helm chart documentation for full details on Kubernetes deployment.

Picocom

> **WARNING**
>
> This document is intended to be used as a guide. Variances in firmware and software versions in local setups may require the sample configuration files provided to be changed. As a result please closely follow the specific users guides of your RU in conjunction with this guide.

Overview

This guide provides further details on connecting the OCUDU CU/DU to an RU using the the O-RAN 7.2 split. Specifically, we discuss the Picocom PC802SCB.

The RU is based on a PC802 Small Cell Development Board (PC802SCB) and an ADI ADRV9029BBCZ 4T4R RFIC subsystem, it is capable of up to 100MHz 4T4R. The PC802SCB is a flexible 5G NR development platform for evaluating the PC802 for different small cell use cases, including as a split 7.2 O-RU. You can read more about it here.

> **WARNING**
>
> The PC802SCB is not a deployment ready commercial grade O-RU, but many commercial O-RUs are based on the PC802 platform.

Configuration

CU/DU

A sample configuration file for the DU can be downloaded from here.

The following excerpt shows how the DU is configured to communicate with the RU:

```
ru_ofh:
t1a_max_cp_dl: 350                                          # Maximum T1a on Control-
Plane for Downlink in microseconds.
t1a_min_cp_dl: 250                                          # Minimum T1a on Control-
Plane for Downlink in microseconds.
t1a_max_cp_ul: 250                                          # Maximum T1a on Control-
Plane for Uplink in microseconds.
t1a_min_cp_ul: 150                                          # Minimum T1a on Control-
Plane for Uplink in microseconds.
```

```
t1a_max_up: 200                                          # Maximum T1a on User-Plane
in microseconds.
t1a_min_up: 80                                           # Minimum T1a on User-Plane
in microseconds.
ta4_max: 300                                             # Maximum Ta4 on User-Plane
in microseconds.
ta4_min: 10                                              # Minimum Ta4 on User-Plane
in microseconds.
is_prach_cp_enabled: false                               # Configures if Control-
Plane messages should be used to receive PRACH messages.
compr_method_ul: bfp                                     # Uplink compression method.
compr_bitwidth_ul: 9                                     # Uplink IQ samples bitwidth
after compression.
compr_method_dl: bfp                                     # Downlink compression
method.
compr_bitwidth_dl: 9                                     # Downlink IQ samples
bitwidth after compression.
compr_method_prach: bfp                                  # PRACH compression method.
compr_bitwidth_prach: 9                                  # PRACH IQ samples bitwidth
after compression.
iq_scaling: 1.0                                          # IQ samples scaling factor
applied before compression, should be a positive value smaller than 10.
cells:
- network_interface: enp1s0f0                          # Ethernet interface name used
to communicate with the RU.
ru_mac_addr: ce:fc:6c:09:a6:cd                       # RU MAC address.
du_mac_addr: 80:61:5f:0d:df:aa                       # DU MAC address.
vlan_tag_cp: 3                                       # VLAN tag value for C-Plane.
vlan_tag_up: 3                                       # VLAN tag value for U-Plane.
prach_port_id: [0]                                   # PRACH eAxC port value.
dl_port_id: [0]                                      # Downlink eAxC port values.
ul_port_id: [0]                                      # Uplink eAxC port values.
```

To expand on this, the following parameters are set in the `cells` field:

- `network_interface` : Network interface used to send the OFH packets.
- `ru_mac_addr` : MAC address of the RU.
- `du_mac_addr` : MAC address of the interface used by the DU for OFH traffic.
- `vlan_tag_up/vlan_tag_cp` : VLAN identifier, should be set to the value configured in the RU/switch settings.

In the `cell_cfg` section of the configuration, the `prach` and `tdd_ul_dl_cfg` fields must also be configured correctly. The following excerpt shows this:

```
prach:
prach_config_index: 159                                  # PRACH configuration index.
prach_root_sequence_index: 1                             # PRACH root sequence index.
zero_correlation_zone: 0                                 # Zero correlation zone.
prach_frequency_start: 0                                 # Offset in PRBs of lowest
PRACH transmission occasion in frequency domain respective to PRB 0.
tdd_ul_dl_cfg:
nof_dl_slots: 7
nof_ul_slots: 2
```

It is important to check these fields as their values may change based on the RU being used.

RU

The first step will be to power-on the board and ensure it is connected correctly to the PC running the CU/DU. Next, log in to the RU via SSH. Once the RU has been accessed correctly via the terminal, it can be configured and connected to the CU/DU and OFH packets can be sent between the two.

> **INFO**
>
> This configuration is relevant to firmware v3.0.0. Other firmware versions may require different configurations. For more information on this reference the relevant documentation from Picocom.

Configuration

The RU must first be configured so that the relevant values match across the DU and RU.

RU OFH related parameters can be found in the file *rf_init_req.json* in this path */home/user/PC-003001-LS/npu/oru_oam/inputs*. Inside this JSON file there is a property called `ecpri_cu_plane_parameters`:

```
"ecpri_cu_plane_params" : {
"is_static_compress" : 1,
"compress_method" : 1,
"compress_iq_width" : 9,
"c_plane_enable": 1,
"ul_padding_enable": 1,
"fhm_via_sectionid_enable": 0,
"cu_mac_check_enable": 0,
"eAxC_id_bit_alloc": 0,
"du_port_id" : 0,
"bandsector_id" : 0,
"cc_id" : 0,
"ru_port_dl": [0, 1, 2, 3],
"ru_port_ul_nonprach": [0, 1, 2, 3],
"ru_port_ul_prach": [0, 1, 2, 3]
},
```

Make sure that Control-Plane is enabled (`"c_plane_enable": 1,`) and that the rest of the parameters match those in the DU configuration.

In this file the Ethernet connection parameters can be found in the property `transport_parameters`:

```
"transport_params" : {
"transport_session_type" : 0,
"vlan_id" : 4000,
"l2_mtu" : 9000,
"odu_mac_addr" : ["0x8C","0x1F","0x64","0xB4","0xC0","0x03"],
```

```
"oru_mac_addr" : ["0x8C","0x1F","0x64","0xB4","0xC1","0xF2"]
},
```

Adjust the `vlan_id`, `odu_mac_addr`, `oru_mac_addr` and `l2_mtu` to match the DU configuration.

RU PRACH related parameters can be found in the file *prach_req.json* in this path */home/user/PC-003001-LS/npu/oru_oam/inputs*. In this file, check the property `prach_cu_plane_params`:

```
"prach_cu_plane_params" : {
"is_static_compress" : 1,
"compress_method" : 1,
"compress_iq_width" : 9,
"c_plane_enable": 1,
"param_source": 0
}
```

Make sure that the settings matches the DU settings.

Another important section of this file is `common_params`:

```
"common_params" : {
"bandwidth" : 100000,
"dl_frame_start_offset" : 0,
"ul_frame_start_offset" : -13021,
"dl_wave_delay" : 918,
"ul_wave_delay" : 1005,
"scs_hz" : 30000,
"cp_type" : 0,
"num_tx_port" : 4,
"num_rx_port" : 4,
"ul_fft_dynamic_scaling_enable" : 1,
"slotnum_txrx_pattern" : 10,
"slotconfig":
["0xf0000000","0xf0000000","0xf0000000","0xf0000000","0xf0000000","0xf0000000","0xf0000000",
"0xf55aa000","0xf5555555","0xf5555555"]
},
```

Make sure that the `slotnum_txrx_pattern`, `cp_type`, `scs_hz` and `bandwidth` parameters matches the DU configuration. The number of antennas can be configured to 4, as the DU will ask the data for the antennas that it configures.

Also pay attention to the `sync_scheme_params` property:

```
"sync_scheme_params" : {
"sync_source" : 3,
"is_pc802_ctrl_vcxo" : 2,
"enable_pc802_sw_filtering_on_1ppsin" : 1,
"pps_tod_sync_threshold" : 100,
"count_pps_insync": 10,
"freq_adj_rate_factor": 4,
```

```
"pps_tod_outsync_threshold": 100,
"gpio_pps_out": "0xFF"
},
```

Property `sync_source` should be configured to use PTP.

If the RSRP detected by the UE is low, is possible to increase it by updating `dl_amplitude_scale` property. Check Picocom documentation for valid values. For reference, OCUDU uses the following by default:

```
"dl_amplitude_scale" : 65535.
```

To enable the output metrics of the RU, make sure OAM is sending the corresponding command to the PC802. The configuration of the messages sent to the PC802 can be found in the file *config.ini* in */home/user/PC-003001-LS/npu/oru_oam*. For example:

```
[MSG_REQ_LIST]
msg_id=0 #rf init req
msg_id=300 #p19 req
msg_id=4 #prach config req
msg_id=1 #start req
msg_id=108 #status
msg_id=109 #measurement status update req
msg_id=110 #measurement counter read req
#msg_id=2 #stop req
#msg_id=65280 #dl scale update
```

Also check the file *measurement_status_update_req_list.json* and *measurement_counter_read_req.json* in *inputs* folder.

As an example the contents of *measurement_counter_read_req.json* should read:

```
{
"measurement_counter_read_req" : {
"measurement_group" : 0,
"object_unit" : 0,
"report_info" : 0
}
}
```

And contents of *measurement_status_update_req_list.json* should read:

```
{
"measurement_status_update_req_list" : {
"num_of_msg" : 2,
"measurement_status_update_msg" : [
{
"measurement_group" : 0,
"measurement_interval" : 1,
```

```
"active" : 1,
"object_unit" : 0,
"report_info" : 0
},
{
"measurement_group" : 1,
"measurement_interval" : 1,
"active" : 1,
"object_unit" : 0,
"report_info" : 0
}
]
}
}
```

Running the RU

Once the RU has been correctly configured, the next step will be to start the RF part of the board. To do this, navigate to the RF folder and run the relevant script with `sudo`:

```
cd PC-003001-LS/npu/RFICDriver/
sudo ./run.sh
```

You should then see the following output or similar:

```
Value is 2, currently Board Type is scb_x3
Run the reset_rfic
Export GPIO 3-2 (418) to userspace.
...
...

adi_board_adrv9025_JesdBringup 1079, adrv9025LinkStatus = 0
ADRV9025 TX deframer status = 18
adi_board_adrv9025_Program_Phase2 1678, adrv9025 RX LinkStatus 4 = 6
main 102
Reg 6a89 read back: 25
RX gain index: 0
RSSI level: 0
```

The kernel module must then be loaded. This can be done from the home directory:

```
sudo insmod pcsc.ko
```

To check that it has been loaded correctly, run the following command:

```
ip a
```

You should see the following output or similar if the kernel module has been loaded correctly:

```
...
8: pcsce0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
link/ether 56:69:eb:b9:65:0b brd ff:ff:ff:ff:ff:ff
```

Next, go to the `ru_oam` folder and start the `oam_manager` process:

```
cd PC-003001-LS/npu/ru_oam/
sudo ./oam_manager
```

The following output should be observed:

```
GIT_BRANCH=master;
GIT_BRANCH=f8233eb84a73f4dea67ec2a118ada4935507603d;
[Initial oam messages sequence sending]:
[##################################################################################################
######### ][100%][-]
<start oam interactive operation>
[oam@operation]#
```

The PTP process now needs to be started on the board. To do this, open two new terminals via SSH. In the first terminal, run the following command:

```
cd PC-003001-LS/npu/linuxptp-3.1.1/
sudo ./ptp4l_run.sh
```

You should then see:

```
ptp4l[23:24: 2.410]: selected /dev/ptp1 as PTP clock
ptp4l[23:24: 2.453]: port 1: INITIALIZING to LISTENING on INIT_COMPLETE
ptp4l[23:24: 2.453]: port 0: INITIALIZING to LISTENING on INIT_COMPLETE
ptp4l[23:24: 2.523]: port 1: new foreign master 000580.fffe.0837cd-20
ptp4l[23:24: 2.789]: selected best master clock 000580.fffe.0837cd
ptp4l[23:24: 2.790]: updating UTC offset to 37
ptp4l[23:24: 2.790]: port 1: LISTENING to UNCALIBRATED on RS_SLAVE
ptp4l[23:24: 3.554]: port 1: UNCALIBRATED to SLAVE on MASTER_CLOCK_SELECTED
ptp4l[23:24: 3.976]: rms 1266036736054421504 max 1688048981730611712 freq -445 +/- 503 delay 19
ptp4l[23:24: 5.101]: rms 153 max 274 freq -315 +/- 263 delay 184 +/- 4
ptp4l[23:24: 6.226]: rms 125 max 164 freq +35 +/- 15 delay 194 +/- 1
...
...
...
ptp4l[23:24:14.101]: rms 2 max 3 freq -39 +/- 3 delay 193 +/- 0
```

In the second terminal, run the `phc2sys` script:

```
cd PC-003001-LS/npu/linuxptp-3.1.1/
sudo ./phc2sys -s /dev/ptp1 -c CLOCK_REALTIME -w -l 7 -n 24 -m
```

This will give the following output:

```
phc2sys[23:25:14.797]: config item (null).clock_servo is 0
phc2sys[23:25:14.798]: config item (null).kernel_leap is 1
phc2sys[23:25:14.798]: config item (null).sanity_freq_limit is 200000000
phc2sys[23:25:14.799]: config item (null).pi_proportional_const is 0.700000
phc2sys[23:25:14.799]: config item (null).pi_integral_const is 0.300000
phc2sys[23:25:14.800]: config item (null).pi_proportional_scale is 0.000000
phc2sys[23:25:14.800]: config item (null).pi_proportional_exponent is -0.300000
phc2sys[23:25:14.800]: config item (null).pi_proportional_norm_max is 0.700000
phc2sys[23:25:14.801]: config item (null).pi_integral_scale is 0.000000
phc2sys[23:25:14.803]: config item (null).pi_integral_exponent is 0.400000
phc2sys[23:25:14.803]: config item (null).pi_integral_norm_max is 0.300000
phc2sys[23:25:14.803]: config item (null).step_threshold is 0.000000
phc2sys[23:25:14.804]: config item (null).first_step_threshold is 0.000020
phc2sys[23:25:14.804]: config item (null).max_frequency is 900000000
phc2sys[23:25:14.804]: config item (null).servo_offset_threshold is 0
phc2sys[23:25:14.805]: config item (null).servo_num_offset_values is 10
phc2sys[23:25:14.805]: PI servo: sync interval 1.000 kp 0.700 ki 0.300000
phc2sys[23:25:14.805]: ioctl PTP_SYS_OFFSET_PRECISE: Operation not supported
phc2sys[23:25:14.806]: ioctl PTP_SYS_OFFSET_EXTENDED: Operation not supported
phc2sys[23:25:14.806]: config item (null).pi_proportional_const is 0.700000
...
...
phc2sys[23:25:14.810]: config item (null).servo_num_offset_values is 10
phc2sys[23:25:14.810]: PI servo: sync interval 1.000 kp 0.700 ki 0.300000
phc2sys[23:25:14.811]: config item (null).domainNumber is 24
phc2sys[23:25:14.811]: config item (null).transportSpecific is 0
phc2sys[23:25:14.811]: config item (null).uds_address is '/var/run/ptp4l'
phc2sys[23:25:15.812]: CLOCK_REALTIME phc offset -9155196481971402 s0 freq          +0 delay
9520
phc2sys[22:31:53.299]: CLOCK_REALTIME phc offset -9155196482007884 s1 freq  -36468 delay
10280
phc2sys[22:31:54.301]: CLOCK_REALTIME phc offset    1539 s2 freq  -34929 delay   8201
...
...
phc2sys[22:31:57.302]: CLOCK_REALTIME phc offset    856 s2 freq  -34994 delay   9361
K_REALTIME phc offset -167 s2 freq  -35761 delay  10161
```

This will need to be left running for 1 to 2 minutes so that the PTP process can stabilize.

Initializing and connecting to the network

Initializing and connecting to the network is done in the same way as outlined in the general 7.2 RU guide.

Initializing the network

The following steps should be taken to initialize the network:

Ensure the RU is started and has PTP lock (see above).

Run the CU/DU, making sure that the PTP sync between the DU and the Falcon switch is successful as previously outlined.

```
sudo ./gnb -c gnb_ru_picocom_scb_tdd_n78_20mhz.yml
```

If the DU connects to the RU successfully, you will see the following output:

```
--== OCUDU gNB (commit 2c67c80) ==--

Connecting to AMF on 127.0.0.5:38412
Initializing the Open Fronthaul Interface for sector#0: ul_compr=[BFP,9], dl_compr=[BFP,9],
prach_compr=[BFP,9], prach_cp_enabled=false, downlink_broadcast=false
Cell pci=1, bw=20 MHz, 4T4R, dl_arfcn=625000 (n78), dl_freq=3375.0 MHz, dl_ssb_arfcn=625056,
ul_freq=3375.0 MHz

==== gNodeB started ===
Type <t> to view trace
```

Once the RU is started and it detects the PTP, a CSV file is created in */home/user/PC-003001-LS/npu/oru_oam*. It prints the KPIs every second with this format, you should see the following or similar:

```
0,TX_TOTAL,2023-03-27T22:30:08+00:00,2023-03-27T22:30:09+00:00,RU,30400
1,TX_TOTAL_C,2023-03-27T22:30:08+00:00,2023-03-27T22:30:09+00:00,RU,0
0,RX_TOTAL,2023-03-27T22:30:08+00:00,2023-03-27T22:30:09+00:00,RU,56000
1,RX_ON_TIME,2023-03-27T22:30:08+00:00,2023-03-27T22:30:09+00:00,RU,0
2,RX_EARLY,2023-03-27T22:30:08+00:00,2023-03-27T22:30:09+00:00,RU,31812
3,RX_LATE,2023-03-27T22:30:08+00:00,2023-03-27T22:30:09+00:00,RU,24188
4,RX_ON_TIME_C,2023-03-27T22:30:08+00:00,2023-03-27T22:30:09+00:00,RU,0
5,RX_EARLY_C,2023-03-27T22:30:08+00:00,2023-03-27T22:30:09+00:00,RU,0
6,RX_LATE_C,2023-03-27T22:30:08+00:00,2023-03-27T22:30:09+00:00,RU,0
```

Connecting to the network

You can now connect a UE to the network. This can be done using e.g. a COTS UE. See the main RU guide for details on this.

VVDN

Overview

This guide provides further details on using OCUDU with O-RUs from VVDN.

Configuration

CU/DU

You can download a sample gNB configuration file that is compatible with the indoor VVDN RU here.

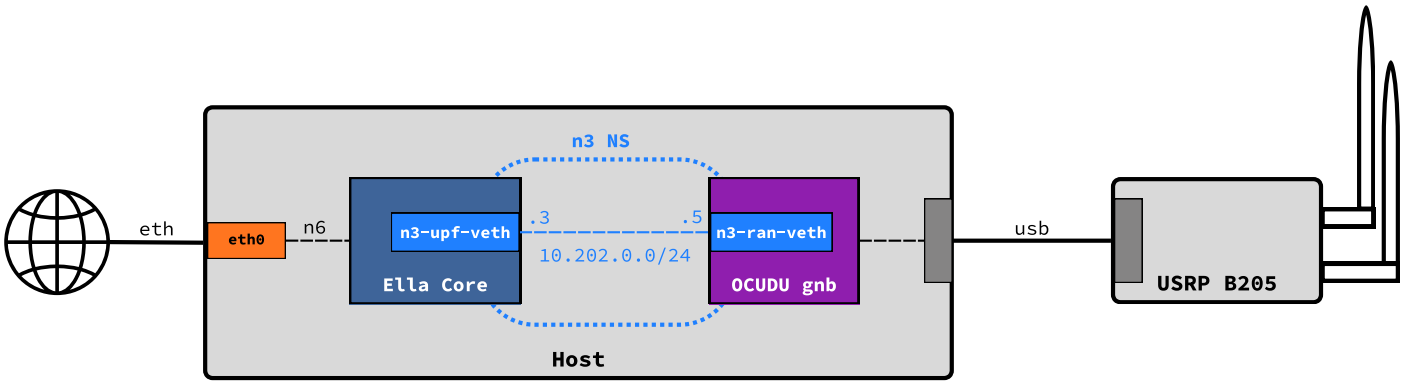
This configuration file will allow you to create a 100MHz SISO TDD cell in band n78.

> **INFO**
>
> A more detailed version of this guide is currently in development. For help with issues and troubleshooting please go to the GitHub Discussions.

Ella Core

Overview

Ella Core is an open-source 5G Core designed for private networks. It ships as a single binary, supports both amd64 and arm64 architectures, and leverages eBPF for high-performance packet processing. Ella Core includes a built-in UI and HTTP API for management and monitoring. OCUDU can either be co-hosted with Ella Core for an all-in-one 5G network, or installed on a different machine. This guide outlines how-to co-host OCUDU and Ella Core.



Resources

- Ella Core Documentation
- Ella Core GitHub Repository

Pre-requisites

- A computer with a Linux-based OS
  - CPU: 4 cores (amd64 or arm64)
  - Memory: 8 GB RAM
  - Storage: 20 GB free disk space
  - Kernel: 6.8 or later
- USRP B205 mini (or any other OCUDU-supported RF front-end)

---

1. Create a network namespace for N3

Connect to the host and create a Linux network namespace `n3ns` for the N3 interface between OCUDU and Ella Core.

```
ip netns add n3ns
ip link add n3-upf-veth type veth peer name n3-ran-veth
ip link set n3-ran-veth netns n3ns
ip addr add 10.202.0.3/24 dev n3-upf-veth
ip -n n3ns addr add 10.202.0.5/24 dev n3-ran-veth
ip -n n3ns link set lo up
ip -n n3ns link set dev n3-ran-veth up
ip link set dev n3-upf-veth up
```

2. Install Ella Core

Install Ella Core:

```
sudo snap install ella-core
sudo snap connect ella-core:network-control
sudo snap connect ella-core:process-control
sudo snap connect ella-core:system-observe
sudo snap connect ella-core:firewall-control
```

> **NOTE**
>
> You can also install Ella Core from source or using Docker. For more details, refer to the Ella Core installation guide.

Edit the configuration file located at `/var/snap/ella-core/common/core.yaml`. Adapt the N2 and N3 interfaces to use `n3-upf-veth`, and N6 to use the system's physical NIC. All configuration parameters are explained in the Ella Core configuration reference. For example:

```
logging:
system:
level: "info"
output: "stdout"
audit:
output: "stdout"
db:
path: "core.db"
interfaces:
n2:
name: "n3-upf-veth"
port: 38412
n3:
name: "n3-upf-veth"
n6:
name: "eth0"
api:
address: "0.0.0.0"
port: 5002
xdp:
attach-mode: "generic"
```

```
telemetry:
enabled: false
```

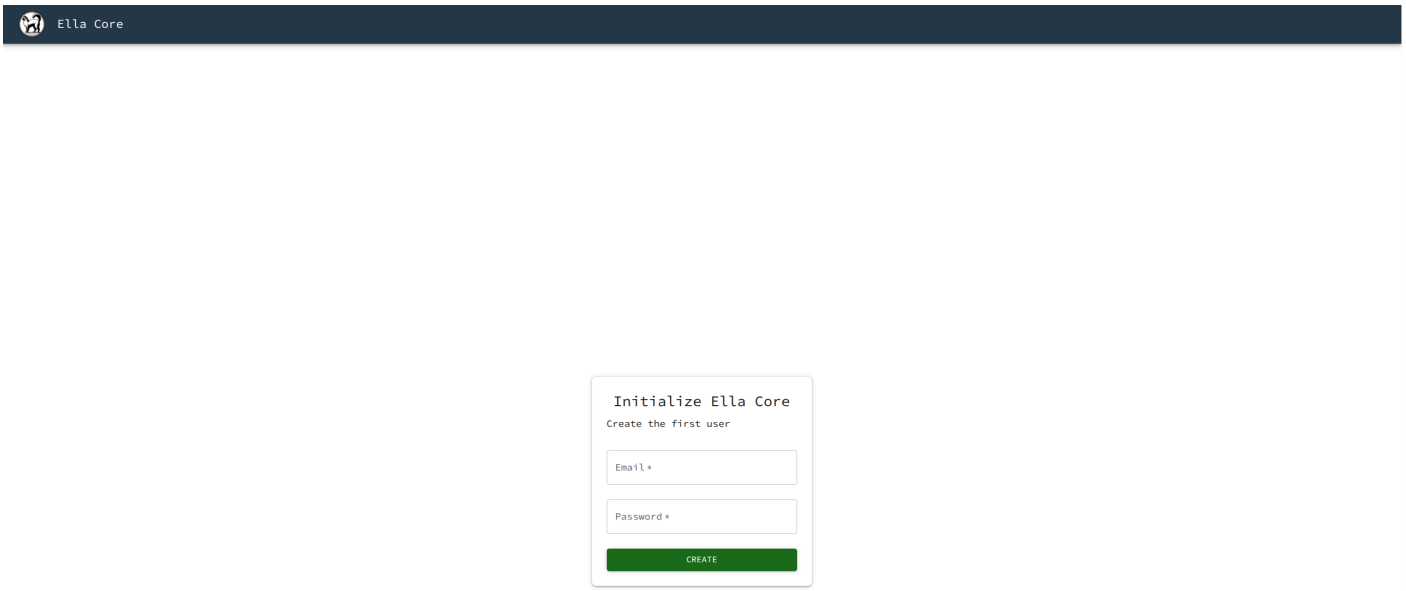
> **NOTE**
>
> If your host's NICs support XDP, edit the XDP attach-mode to `native` for better performance.

Start Ella Core:

```
sudo snap start ella-core --enable
```

3. Initialize Ella Core

Open your browser at the system's IP address on port 5002, you should see the initialization screen.



Navigate to the Operator tab and set your MCC and MNC. Here we will be using 999–01.

Navigate to the Subscribers tab and create a new subscriber. You can click “Generate” to get randomly generated values for IMSI, Key, and OPc.

4. Install OCUDU

Install OCUDU using the OCUDU installation guide.

In the OCUDU configuration file, use the addresses we defined when creating the `n3ns` namespace and the same PLMN ID configured in Ella Core. For example:

```
cu_cp:
amf:
addr: 10.202.0.3
port: 38412
bind_addr: 10.202.0.5
supported_tracking_areas:
- tac: 1
plmn_list:
- plmn: "99901"
tai_slice_support_list:
- sst: 1
inactivity_timer: 300
security:
nea_pref_list: nea2,nea1
nia_pref_list: nia2,nia1
cu_up:
ngu:
socket:
- bind_addr: 10.202.0.5
ru_sdr:
device_driver: uhd
device_args: type=b200
clock: internal
srate: 23.04
tx_gain: 80
rx_gain: 40

cell_cfg:
dl_arfcn: 665000
band: 77
channel_bandwidth_MHz: 20
common_scs: 30
plmn: "99901"
tac: 1
pdcch:
dedicated:
ss2_type: common
dci_format_0_1_and_1_1: false
prach:
prach_config_index: 160
pdsch:
mcs_table: qam64
pusch:
mcs_table: qam64

log:
filename: /tmp/gnb.log
all_level: info

pcap:
mac_enable: enable
mac_filename: /tmp/gnb_mac.pcap
ngap_enable: enable
ngap_filename: /tmp/gnb_ngap.pcap
```

Start OCUDU's gnb in the `n3ns` namespace:

```
sudo ip netns exec n3ns ./gnb -c gnb.yaml
```

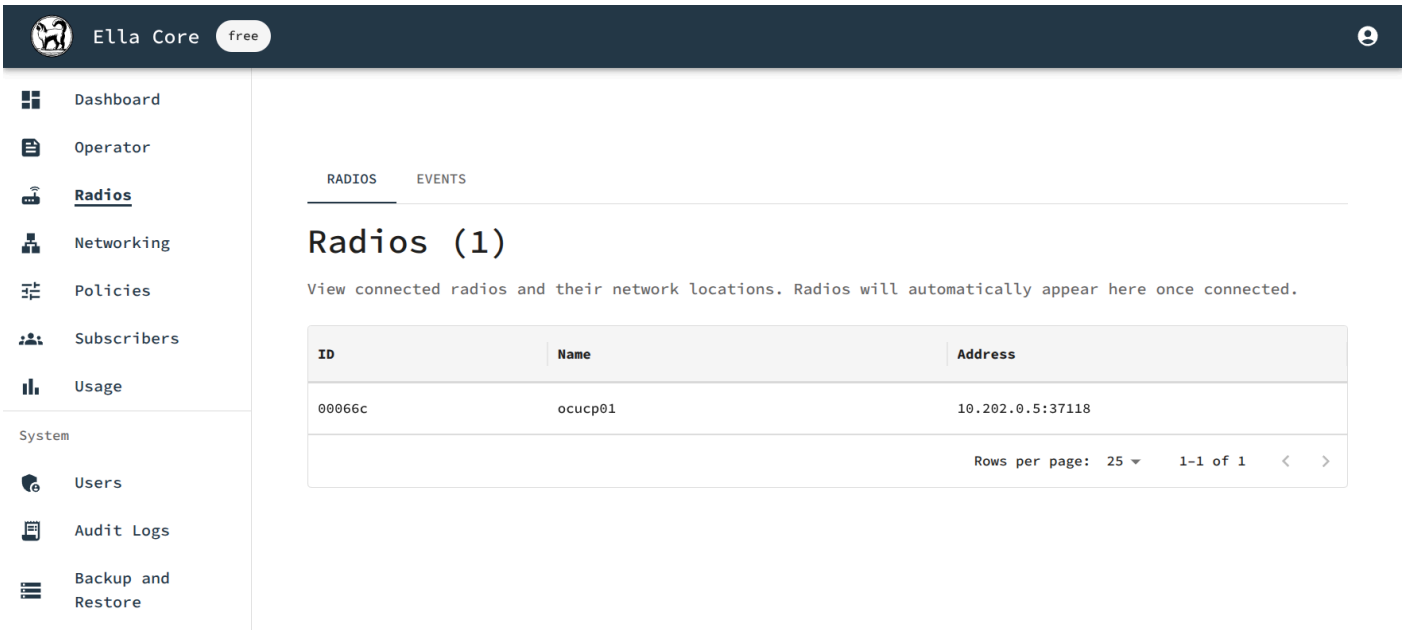
You should see the gnb detecting the B205 mini and connecting to Ella Core:

```
--== OCUDU gNB (commit d1ca6d2744) ==--

2026-02-16T18:34:35.963576 [GNB     ] [I] Built in Release mode using commit d1ca6d2744 on
branch dev
Lower PHY in dual baseband executor mode.
Available radio types: uhd.
[INFO] [UHD] linux; GNU C++ version 13.2.0; Boost_108300; UHD_4.6.0.0+ds1-5.1ubuntu0.24.04.1
Making USRP object with args 'type=b200'
[INFO] [LOGGING] Fastpath logging disabled at runtime.
[INFO] [B200] Detected Device: B205mini
[INFO] [B200] Operating over USB 3.
[INFO] [B200] Initialize CODEC control...
[INFO] [B200] Initialize Radio control...
[INFO] [B200] Performing register loopback test...
[INFO] [B200] Register loopback test passed
[INFO] [B200] Setting master clock rate selection to 'automatic'.
[INFO] [B200] Asking for clock rate 16.000000 MHz...
[INFO] [B200] Actually got clock rate 16.000000 MHz.
[INFO] [MULTI_USRP] Setting master clock rate selection to 'manual'.
[INFO] [B200] Asking for clock rate 23.040000 MHz...
[INFO] [B200] Actually got clock rate 23.040000 MHz.
Cell pci=1, bw=20 MHz, 1T1R, dl_arfcn=665000 (n77), dl_freq=3975 MHz, dl_ssb_arfcn=664704,
ul_freq=3975 MHz

N2: Connection to AMF on 10.202.0.3:38412 completed
Remote control server listening on 0.0.0.0:8001
==== gNB started ===
```

You should see the radio in Ella Core's UI:



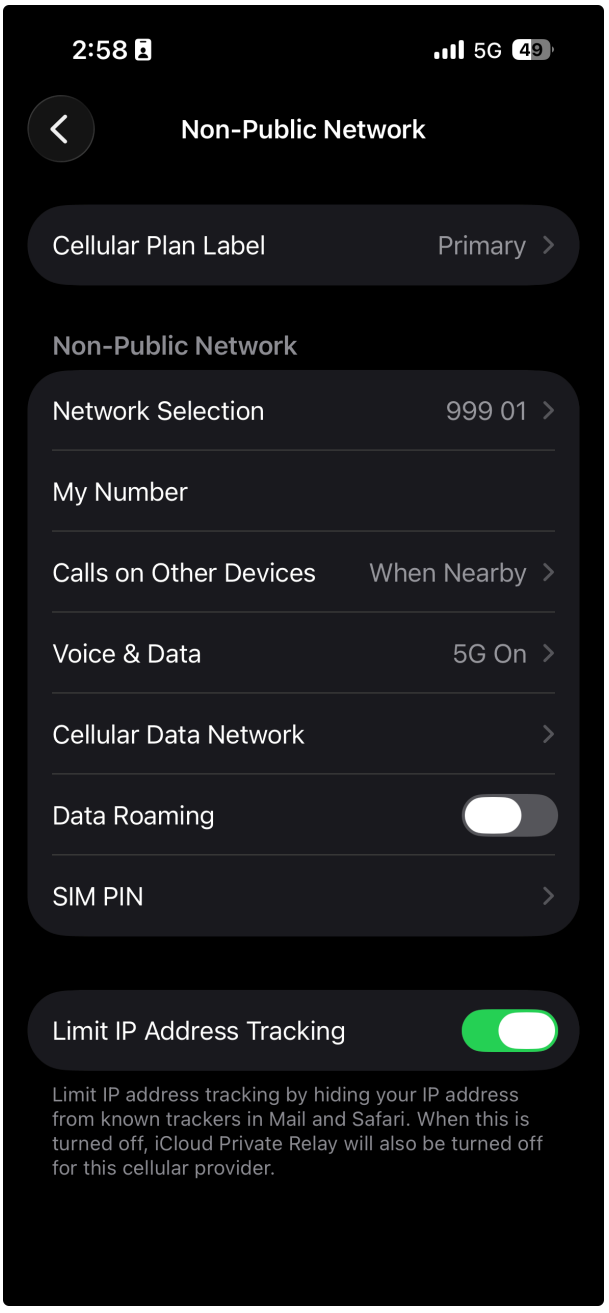
5. Connect a 5G device

Burn the SIM card with the same information you used to create the subscriber in Ella Core.

Insert the SIM card into the phone.

Configure the phone with “internet” as its APN and ensure it is set to “5G Standalone”.

The phone should now connect to the 5G network.



Conclusion

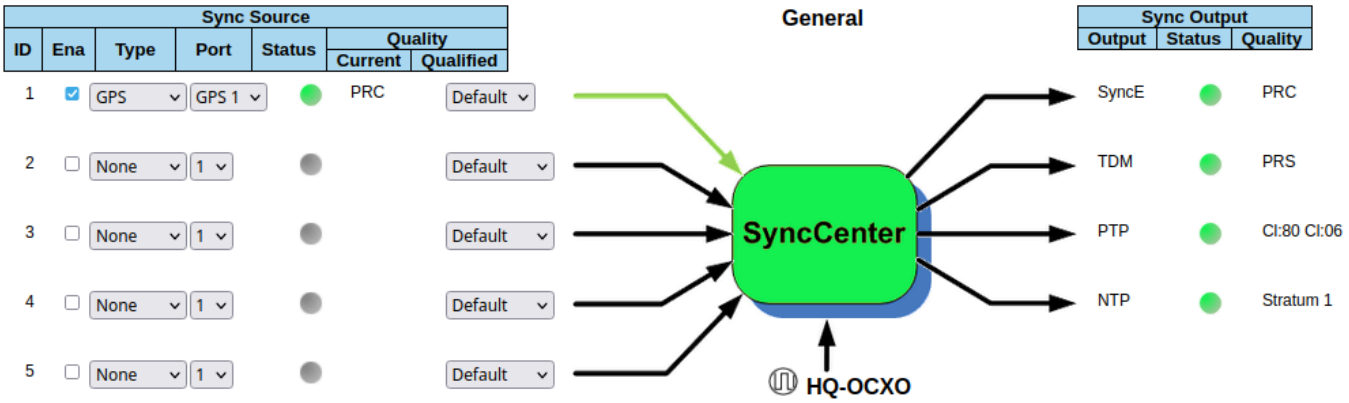
You have successfully set up a 5G network using OCUDU and Ella Core and connected a 5G device to it!

Falcon-RX Switch

The Falcon-RX/812/G Switch is a 5G xHaul timing-aware O-RAN switch & PTP grandmaster. This is used to provide both clocking and timing synchronization to both the DU and RU.

SyncCenter

The switch must be connected to an external clock source to ensure the PTP grandmaster is synchronized correctly. Once connected it is important to check that the GPS has been locked correctly and an accurate clock source is being provided. In this example a GPS reference is used.



To do this, navigate to the FalconRX configuration GUI and go to *Configuration > Timing > SyncCenter* and select `GPS` as the `Sync Source Type`. Once this is done, wait for the GPS to lock and synchronize correctly. The SyncCenter will display green once it has successfully locked to the GPS signal. This is shown in the above image.

PTP Clocks

Once the PTP grandmaster is successfully synchronized it must be configured correctly for use with the DU and RU.

PTP Clock Configuration

Delete	Clock Instance	HW Domain	Device Type	Profile	Clock Description
☐	0	0	Mastronly	G8275.1	Benetel R550
☐	1	0	Mastronly	G8275.1	Foxconn RU

Add New PTP Clock

Apply Reset

PTP Status SyncCenter Config

First, go to *Configuration > Timing > PTP* and add a new PTP Clock. Select `Device Type: Master Only` and `Profile: G8275.1`. This is shown in the above image. After adding the `PTP clock`, click on the Clock Instance that you want to edit.

PTP Clock's Configuration and Status

Clock Type and Profile

Clock Instance	HW Domain	Device Type	Profile	Apply Profile Defaults
0	0	Mastronly	G8275.1	Apply

Port Enable and Configuration

Port Enable																			Configuration
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	Ports Configuration
☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☑	☑	☑	☑	☑	☑	☑	☑	☐	

Local Clock Current Time

PTP Time
2023-05-25T11:56:08+00:00 721,263,893

Clock Current DataSet

stpRm	Offset From Master	Mean Path Delay
0	0.000,000,000,000	0.000,000,000,000

Clock Parent DataSet

Parent Port ID	port	PStat	Var	Rate	GrandMaster ID	GrandMaster Clock Quality	Pri1	Pri2
00:05:80:ff:fe:08:37:cd	0	False	0	0	00:05:80:ff:fe:08:37:cd	Cl:006 Ac:100 ns Va:20061	128	128

Clock Default DataSet

Device Type	One-Way	2 Step Flag	Ports	Clock Identity	Dom	Clock Quality
Mastronly	False	False	21	00:05:80:ff:fe:08:37:cd	24	Cl:006 Ac:100 ns Va:20061

Pri1	Pri2	Local Prio	Protocol	VID	PCP	DSCP
128	128	128	Ethernet	1588	0	0

Master Enable on State

Free Run	Lock Acquisition	Locked	Holdover	Holdover Recovery
Enable	Enable	Enable	Enable	Enable

Quality Override

Class	Accuracy	Variance
☐ 0	☐ 0	☐ 0

Clock Time Properties DataSet

UtcOffset	Valid	leap59	leap61	Time Trac	Freq Trac	ptp Time Scale	Time Source
37	True	False	False	True	True	True	32

Leap Pending	Leap Date	Leap Type
False	1970-01-01	leap59

Apply Reset

PTP Global PTP Clock Status

Once you have selected the `Clock Instance` you want to edit, set the `VLAN ID` to `1588` and activate all ports that you want to serve with PTP. From now on the PTP is sent with VLAN ID 1588.

You should now save your configuration.

VLAN

Next, the VLANs must be configured correctly so as to allow the DU and RU to receive the PTP sync from the grandmaster.

Global VLAN Configuration

Allowed Access VLANs	1,2
Ethertype for Custom S-ports	88A8

Port VLAN Configuration

Port	Mode	Port VLAN	Port Type	Ingress Filtering	Ingress Acceptance	Egress Tagging	Allowed VLANs	Forbidden VLANs
*	<>	1588	<>	☑	<>	<>	1,2,1588	
1	Trunk	1588	C-Port	☑	Tagged and Untagged	Untag Port VLAN	1,2,1588	
2	Trunk	1588	C-Port	☑	Tagged and Untagged	Untag Port VLAN	1,2,1588	
3	Trunk	1588	C-Port	☑	Tagged and Untagged	Untag Port VLAN	1,2,1588	
4	Trunk	1588	C-Port	☑	Tagged and Untagged	Untag Port VLAN	1,2,1588	
5	Trunk	1588	C-Port	☑	Tagged and Untagged	Untag Port VLAN	1,2,1588	
6	Trunk	1588	C-Port	☑	Tagged and Untagged	Untag Port VLAN	1,2,1588	
7	Trunk	1588	C-Port	☑	Tagged and Untagged	Untag Port VLAN	1,2,1588	
8	Trunk	1588	C-Port	☑	Tagged and Untagged	Untag Port VLAN	1,2,1588	
9	Trunk	1588	C-Port	☑	Tagged and Untagged	Untag Port VLAN	1,2,1588	
10	Trunk	1588	C-Port	☑	Tagged and Untagged	Untag Port VLAN	1,2,1588	
11	Trunk	1588	C-Port	☑	Tagged and Untagged	Untag Port VLAN	1,2,1588	
12	Trunk	1588	C-Port	☑	Tagged and Untagged	Untag Port VLAN	1,2,1588	
13	Trunk	1588	C-Port	☑	Tagged and Untagged	Untag Port VLAN	1,2,1588	
14	Trunk	1588	C-Port	☑	Tagged and Untagged	Untag Port VLAN	1,2,1588	
15	Trunk	1588	C-Port	☑	Tagged and Untagged	Untag Port VLAN	1,2,1588	
16	Trunk	1588	C-Port	☑	Tagged and Untagged	Untag Port VLAN	1,2,1588	
17	Trunk	1588	C-Port	☑	Tagged and Untagged	Untag Port VLAN	1,2,1588	
18	Trunk	1588	C-Port	☑	Tagged and Untagged	Untag Port VLAN	1,2,1588	
19	Trunk	1588	C-Port	☑	Tagged and Untagged	Untag Port VLAN	1,2,1588	
20	Trunk	1588	C-Port	☑	Tagged and Untagged	Untag Port VLAN	1,2,1588	
21	Access	1	C-Port	☑	Tagged and Untagged	Untag All	1	

Apply Reset

Go to *Configuration > VLANs > Configuration* to correctly configure the VLAN settings. First, set `Allowed Access VLANs:` as `1,2`. Next, configure the ports you want to use as `Trunk` ports, set the `Port VLAN` as `1588`, and set `Egress Tagging` as `Untag Port VLAN`. In the `Allowed VLANs` field you can set a range or specify specific VLANs. For example, here we are specifying `1,2,1588`. You **must** include `1588` otherwise the DU and RU will not correctly receive the PTP sync.

Meinberg

Meinberg produces PTP Grandmaster clocks suitable for providing timing and synchronization in O-RAN fronthaul networks. In general, any Meinberg device with announced PTP support should be compatible — see the Meinberg website for the full list of supported devices.

The following devices have been tested in-house with OCUDU:

- **Meinberg MicroSync HR** — verified as PTP Grandmaster in combination with MikroTik switches acting as PTP Boundary Clocks.

MikroTik

MikroTik produces a range of switches with PTP support suitable for use as timing-aware O-RAN fronthaul switches. In general, any MikroTik device with announced PTP support should be compatible - see the MikroTik wiki for the full list.

The following devices have been tested in-house with OCUDU:

- **CRS326-24S+2Q+**
- **CRS510-8XS-2XQ-IN**

Netgear

Netgear produces a range of switches with PTP support suitable for use as timing-aware O-RAN fronthaul switches. In general, any Netgear device with announced PTP support should be compatible — see the Netgear website for the full list of supported devices.

The following devices have been tested in-house with OCUDU:

- **Netgear M4300-8X8F**