



# DPDK

DATA PLANE DEVELOPMENT KIT

## Baseband Device Drivers

*Release 18.08.0*

August 31, 2018

## CONTENTS

<b>1</b>	<b>BBDEV null Poll Mode Driver</b>	<b>1</b>
1.1	Limitations . . . . .	1
1.2	Installation . . . . .	1
1.3	Initialization . . . . .	1
<b>2</b>	<b>SW Turbo Poll Mode Driver</b>	<b>2</b>
2.1	Features . . . . .	2
2.2	Limitations . . . . .	2
2.3	Installation . . . . .	3
2.4	Initialization . . . . .	4

## BBDEV NULL POLL MODE DRIVER

The (**baseband\_null**) is a bbdev poll mode driver which provides a minimal implementation of a software bbdev device. As a null device it does not modify the data in the mbuf on which the bbdev operation is to operate and it only works for operation type `RTE_BBDEV_OP_NONE`.

When a burst of mbufs is submitted to a *bbdev null PMD* for processing then each mbuf in the burst will be enqueued in an internal buffer ring to be collected on a dequeue call.

### 1.1 Limitations

- In-place operations for Turbo encode and decode are not supported

### 1.2 Installation

The *bbdev null PMD* is enabled and built by default in both the Linux and FreeBSD builds.

### 1.3 Initialization

To use the PMD in an application, user must:

- Call `rte_vdev_init("baseband_null")` within the application.
- Use `--vdev="baseband_null"` in the EAL options, which will call `rte_vdev_init()` internally.

The following parameters (all optional) can be provided in the previous two calls:

- `socket_id`: Specify the socket where the memory for the device is going to be allocated (by default, `socket_id` will be the socket where the core that is creating the PMD is running on).
- `max_nb_queues`: Specify the maximum number of queues in the device (default is `RTE_MAX_LCORE`).

#### 1.3.1 Example:

```
./test-bbdev.py -e="--vdev=baseband_null,socket_id=0,max_nb_queues=8"
```

## SW TURBO POLL MODE DRIVER

The SW Turbo PMD (**baseband\_turbo\_sw**) provides a poll mode bbdev driver that utilizes Intel optimized libraries for LTE Layer 1 workloads acceleration. This PMD supports the functions: Turbo FEC, Rate Matching and CRC functions.

### 2.1 Features

SW Turbo PMD has support for the following capabilities:

For the encode operation:

- RTE\_BBDEV\_TURBO\_CRC\_24A\_ATTACH
- RTE\_BBDEV\_TURBO\_CRC\_24B\_ATTACH
- RTE\_BBDEV\_TURBO\_RATE\_MATCH
- RTE\_BBDEV\_TURBO\_RV\_INDEX\_BYPASS

For the decode operation:

- RTE\_BBDEV\_TURBO\_SUBBLOCK\_DEINTERLEAVE
- RTE\_BBDEV\_TURBO\_CRC\_TYPE\_24B
- RTE\_BBDEV\_TURBO\_POS\_LLR\_1\_BIT\_IN
- RTE\_BBDEV\_TURBO\_NEG\_LLR\_1\_BIT\_IN
- RTE\_BBDEV\_TURBO\_DEC\_TB\_CRC\_24B\_KEEP
- RTE\_BBDEV\_TURBO\_EARLY\_TERMINATION

### 2.2 Limitations

- In-place operations for Turbo encode and decode are not supported

## 2.3 Installation

### 2.3.1 FlexRAN SDK Download

To build DPDK with the *baseband\_turbo\_sw* PMD the user is required to download the export controlled FlexRAN SDK Libraries. An account at [Intel Resource Design Center](#) needs to be registered.

Once registered, the user needs to log in, and look for *Intel FlexRAN Software Release Package -1-6-0* to download or directly through this [link](#).

After download is complete, the user needs to unpack and compile on their system before building DPDK.

The following table maps DPDK versions with past FlexRAN SDK releases:

Table 2.1: DPDK and FlexRAN SDK releases compliance

DPDK version	FlexRAN SDK release
18.02	1.3.0
18.05	1.4.0
18.08	1.6.0

### 2.3.2 FlexRAN SDK Installation

The following are pre-requisites for building FlexRAN SDK Libraries:

1. An AVX2 supporting machine
2. CentOS Linux release 7.2.1511 (Core) operating system
3. Intel ICC 18.0.1 20171018 compiler installed

The following instructions should be followed in this exact order:

1. Set the environment variables:

```
source <path-to-icc-compiler-install-folder>/linux/bin/compilervars.sh intel64 -platform
```

2. Extract the flexran-1-6-0-tar.gz.zip package:

```
unzip flexran-1-6-0-tar.gz.zip
tar xvzf flexran-1-6-0-tar.gz -C FlexRAN-1.6.0/
```

3. Run the SDK extractor script and accept the license:

```
cd <path-to-workspace>/FlexRAN-1.6.0/
./SDK-R1.6.0.sh
```

4. Generate makefiles based on system configuration:

```
cd <path-to-workspace>/FlexRAN-1.6.0/SDK-R1.6.0/sdk/
./create-makefiles-linux.sh
```

5. A build folder is generated in this form build-<ISA>-<CC>, enter that folder and install:

```
cd build-avx2-icc/
make && make install
```

## 2.4 Initialization

In order to enable this virtual bbdev PMD, the user must:

- Build the `FLEXRAN` SDK libraries (explained in Installation section).
- Export the environmental variables `FLEXRAN_SDK` to the path where the FlexRAN SDK libraries were installed. And `DIR_WIRELESS_SDK` to the path where the libraries were extracted.

Example:

```
export FLEXRAN_SDK=<path-to-workspace>/FlexRAN-1.6.0/SDK-R1.6.0/sdk/build-avx2-icc/install
export DIR_WIRELESS_SDK=<path-to-workspace>/FlexRAN-1.6.0/SDK-R1.6.0/sdk/
```

- Set `CONFIG_RTE_LIBRTE_PMD_BBDEV_TURBO_SW=y` in DPDK common configuration file `config/common_base`.

To use the PMD in an application, user must:

- Call `rte_vdev_init("baseband_turbo_sw")` within the application.
- Use `--vdev="baseband_turbo_sw"` in the EAL options, which will call `rte_vdev_init()` internally.

The following parameters (all optional) can be provided in the previous two calls:

- `socket_id`: Specify the socket where the memory for the device is going to be allocated (by default, `socket_id` will be the socket where the core that is creating the PMD is running on).
- `max_nb_queues`: Specify the maximum number of queues in the device (default is `RTE_MAX_LCORE`).

### 2.4.1 Example:

```
./test-bbdev.py -e="--vdev=baseband_turbo_sw,socket_id=0,max_nb_queues=8" \
-c validation -v ./turbo*_default.data
```