DPDK

DATA PLANE DEVELOPMENT KIT

Baseband Device Drivers

Release 18.05.1

September 07, 2018

CONTENTS

CHAPTER

ONE

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"
```

CHAPTER

TWO

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_4_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

2.3.2 FlexRAN SDK Installation

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

1. An AVX2 supporting machine
2. Windriver TS 2 or CentOS 7 operating systems
3. Intel ICC 17.0.3 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 -platf
   ```

2. Extract the `576288-576288-intel-swa-sw-flexran-release-package-r1-4-0.zip` package, then run the SDK extractor script and accept the license:

   ```
   cd <path-to-workspace>/FlexRAN-1.4.0/
   ./SDK-R1.4.0.sh
   ```

3. Generate makefiles based on system configuration:

   ```
   cd <path-to-workspace>/FlexRAN-1.4.0/SDK-R1.4.0/sdk/
   ./create-makefiles-linux.sh
   ```

4. If build as a shared library is required, then add `"-fPIC"` option to file `<path-to-workspace>/FlexRAN-1.4.0/SDK-R1.4.0/sdk/cmake/intel-compile-options` as follows:

   ```
   add_compile_options("-fPIC")
   ```

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

   ```
   cd build-avx2-icc/
   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-1.4.0/SDK-R1.4.0/sdk/build-avx2-icc/install
export DIR_WIRELESS_SDK=<path-to-workspace>/FlexRAN-1.4.0/SDK-R1.4.0/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 ./test_vectors/bbdev_vector_t?_default.data
```