DPDK

DATA PLANE DEVELOPMENT KIT

Platform Specific Guides

Release 19.05.0

May 13, 2019

1 **Mellanox BlueField Board Support Package** **2**
- 1.1 Supported BlueField family SoCs 2
- 1.2 Supported BlueField Platforms 2
- 1.3 Common Offload HW Drivers 2
- 1.4 Steps To Setup Platform 2
- 1.5 Compile DPDK 3

2 **NXP QorIQ DPAA Board Support Package** **5**
- 2.1 Supported DPAA SoCs 5
- 2.2 Common Offload HW Block Drivers 5
- 2.3 Steps To Setup Platform 5

3 **NXP QorIQ DPAA2 Board Support Package** **7**
- 3.1 Supported DPAA2 SoCs 7
- 3.2 Common Offload HW Block Drivers 7
- 3.3 Steps To Setup Platform 7

4 **OCTEON TX Board Support Package** **9**
- 4.1 Common Offload HW Block Drivers 9
- 4.2 Steps To Setup Platform 9
- 4.3 Setup Platform Using OCTEON TX SDK 10

The following are platform specific guides and setup information.

CHAPTER

ONE

MELLANOX BLUEFIELD BOARD SUPPORT PACKAGE

This document has information about steps to setup Mellanox BlueField platform and common offload HW drivers of **Mellanox BlueField** family SoC.

1.1 Supported BlueField family SoCs

- BlueField

1.2 Supported BlueField Platforms

- BlueField SmartNIC
- BlueField Reference Platforms
- BlueField Controller Card

1.3 Common Offload HW Drivers

1. **NIC Driver**

   See `../nics/mlx5` for Mellanox mlx5 NIC driver information.

2. **Cryptodev Driver**

   This is based on the crypto extension support of armv8. See `../cryptodevs/armv8` for armv8 crypto driver information.

Note: BlueField has a variant having no armv8 crypto extension support.

1.4 Steps To Setup Platform

Toolchains, OS and drivers can be downloaded and installed individually from the Web. But it is recommended to follow instructions at Mellanox BlueField Software Website.

1.5 Compile DPDK

DPDK can be compiled either natively on BlueField platforms or cross-compiled on an x86 based platform.

1.5.1 Native Compilation

Refer to `../nics/mlx5` for prerequisites. Either Mellanox OFED/EN or rdma-core library with corresponding kernel drivers is required.

make build

```
make config T=arm64-bluefield-linuxapp-gcc
make -j
```

meson build

```
meson build
ninja -C build
```

1.5.2 Cross Compilation

Refer to `../linux_gsg/cross_build_dpdk_for_arm64` to install the cross toolchain for ARM64. Base on that, additional header files and libraries are required:

- libibverbs
- libmnl
- libmlx5
- libnl-3
- libnl-route-3

Such header files and libraries can be cross-compiled and installed on to the cross toolchain directory like depicted in arm_cross_build_getting_the_prerequisite_library, but those can also be simply copied from the filesystem of a working BlueField platform. The following script can be run on a BlueField platform in order to create a supplementary tarball for the cross toolchain.

```
mkdir -p aarch64-linux-gnu/libc
pushd $PWD
cd aarch64-linux-gnu/libc

# Copy libraries
mkdir -p lib64
cp -a /lib64/libibverbs* lib64/
cp -a /lib64/libmnl* lib64/
cp -a /lib64/libmlx5* lib64/
cp -a /lib64/libnl-3* lib64/
cp -a /lib64/libnl-route-3* lib64/

# Copy header files
mkdir -p usr/include/infiniband
mkdir -p usr/include/libmnl
cp -a /usr/include/infiniband/ib_user_ioctl_verbs.h usr/include/infiniband/
cp -a /usr/include/infiniband/mlx5*.h usr/include/infiniband/
cp -a /usr/include/infiniband/tm_types.h usr/include/infiniband/
cp -a /usr/include/infiniband/verbs*.h usr/include/infiniband/
cp -a /usr/include/libmnl/libmnl.h usr/include/libmnl/

# Create supplementary tarball
popd
tar cf aarch64-linux-gnu-mlx.tar aarch64-linux-gnu/
```

Then, untar the tarball at the cross toolchain directory on the x86 host.

```
cd $(dirname $(which aarch64-linux-gnu-gcc))/..
tar xf aarch64-linux-gnu-mlx.tar
```

make build

```
make config T=arm64-bluefield-linuxapp-gcc
make -j CROSS=aarch64-linux-gnu- CONFIG_RTE_KNI_KMOD=n CONFIG_RTE_EAL_IGB_UIO=n
```

meson build

```
meson build --cross-file config/arm/arm64_bluefield_linux_gcc
ninja -C build
```

CHAPTER

TWO

NXP QORIQ DPAA BOARD SUPPORT PACKAGE

This doc has information about steps to setup QorIQ dpaa based layerscape platform and information about common offload hw block drivers of **NXP QorIQ DPAA** SoC family.

2.1 Supported DPAA SoCs

- LS1046A/LS1026A
- LS1043A/LS1023A

More information about SoC can be found at NXP Official Website.

2.2 Common Offload HW Block Drivers

1. **Nics Driver**

   See `../nics/dpaa` for NXP dpaa nic driver information.

2. **Cryptodev Driver**

   See `../cryptodevs/dpaa_sec` for NXP dpaa cryptodev driver information.

3. **Eventdev Driver**

   See `../eventdevs/dpaa` for NXP dpaa eventdev driver information.

2.3 Steps To Setup Platform

There are four main pre-requisites for executing DPAA PMD on a DPAA compatible board:

1. **ARM 64 Tool Chain**

   For example, the *aarch64* Linaro Toolchain.

2. **Linux Kernel**

   It can be obtained from NXP's Github hosting.

3. **Rootfile system**

   Any *aarch64* supporting filesystem can be used. For example, Ubuntu 16.04 LTS (Xenial) or 18.04 (Bionic) userland which can be obtained from here.

4. **FMC Tool**

   Before any DPDK application can be executed, the Frame Manager Configuration Tool (FMC) need to be executed to set the configurations of the queues. This includes the queue state, RSS and other policies. This tool can be obtained from NXP (Freescale) Public Git Repository.

   This tool needs configuration files which are available in the *DPDK Extra Scripts*, described below for DPDK usages.

As an alternative method, DPAA PMD can also be executed using images provided as part of SDK from NXP. The SDK includes all the above prerequisites necessary to bring up a DPAA board.

The following dependencies are not part of DPDK and must be installed separately:

- **NXP Linux SDK**

  NXP Linux software development kit (SDK) includes support for family of QorIQ® ARM-Architecture-based system on chip (SoC) processors and corresponding boards.

  It includes the Linux board support packages (BSPs) for NXP SoCs, a fully operational tool chain, kernel and board specific modules.

  SDK and related information can be obtained from: NXP QorIQ SDK.

- **DPDK Extra Scripts**

  DPAA based resources can be configured easily with the help of ready scripts as provided in the DPDK Extra repository.

  DPDK Extras Scripts.

Currently supported by DPDK:

- NXP SDK **2.0+** (preferred: LSDK 18.09).
- Supported architectures: **arm64 LE**.
- Follow the DPDK Getting Started Guide for Linux to setup the basic DPDK environment.

CHAPTER

THREE

NXP QORIQ DPAA2 BOARD SUPPORT PACKAGE

This doc has information about steps to setup NXP QorIQ DPAA2 platform and information about common offload hw block drivers of **NXP QorIQ DPAA2** SoC family.

3.1 Supported DPAA2 SoCs

- LX2160A
- LS2084A/LS2044A
- LS2088A/LS2048A
- LS1088A/LS1048A

More information about SoC can be found at NXP Official Website.

3.2 Common Offload HW Block Drivers

1. **Nics Driver**

   See `../nics/dpaa2` for NXP dpaa2 nic driver information.

2. **Cryptodev Driver**

   See `../cryptodevs/dpaa2_sec` for NXP dpaa2 cryptodev driver information.

3. **Eventdev Driver**

   See `../eventdevs/dpaa2` for NXP dpaa2 eventdev driver information.

4. **Rawdev AIOP CMDIF Driver**

   See `../rawdevs/dpaa2_cmdif` for NXP dpaa2 AIOP command interface driver information.

5. **Rawdev QDMA Driver**

   See `../rawdevs/dpaa2_qdma` for NXP dpaa2 QDMA driver information.

3.3 Steps To Setup Platform

There are four main pre-requisites for executing DPAA2 PMD on a DPAA2 compatible board:

1. **ARM 64 Tool Chain**

   For example, the *aarch64* Linaro Toolchain.

2. **Linux Kernel**

   It can be obtained from NXP's Github hosting.

3. **Rootfile system**

   Any *aarch64* supporting filesystem can be used. For example, Ubuntu 16.04 LTS (Xenial) or 18.04 (Bionic) userland which can be obtained from here.

4. **Resource Scripts**

   DPAA2 based resources can be configured easily with the help of ready scripts as provided in the DPDK Extra repository.

As an alternative method, DPAA2 PMD can also be executed using images provided as part of SDK from NXP. The SDK includes all the above prerequisites necessary to bring up a DPAA2 board.

The following dependencies are not part of DPDK and must be installed separately:

- **NXP Linux SDK**

  NXP Linux software development kit (SDK) includes support for family of QorIQ® ARM-Architecture-based system on chip (SoC) processors and corresponding boards.

  It includes the Linux board support packages (BSPs) for NXP SoCs, a fully operational tool chain, kernel and board specific modules.

  SDK and related information can be obtained from: NXP QorIQ SDK.

- **DPDK Extra Scripts**

  DPAA2 based resources can be configured easily with the help of ready scripts as provided in the DPDK Extra repository.

  DPDK Extras Scripts.

Currently supported by DPDK:

- NXP SDK **2.0+** (preferred: LSDK 19.03).
- MC Firmware version **10.14.0** and higher.
- Supported architectures: **arm64 LE**.
- Follow the DPDK Getting Started Guide for Linux to setup the basic DPDK environment.

CHAPTER

FOUR

OCTEON TX BOARD SUPPORT PACKAGE

This doc has information about steps to setup OCTEON TX platform and information about common offload hw block drivers of **Cavium OCTEON TX** SoC family.

More information about SoC can be found at Cavium, Inc Official Website.

4.1 Common Offload HW Block Drivers

1. **Crypto Driver** See `../cryptodevs/octeontx` for octeontx crypto driver information.
2. **Eventdev Driver** See `../eventdevs/octeontx` for octeontx ssovf eventdev driver information.
3. **Mempool Driver** See `../mempool/octeontx` for octeontx fpavf mempool driver information.

4.2 Steps To Setup Platform

There are three main pre-prerequisites for setting up Platform drivers on OCTEON TX compatible board:

1. **OCTEON TX Linux kernel PF driver for Network acceleration HW blocks**

   The OCTEON TX Linux kernel drivers (includes the required PF driver for the Platform drivers) are available on Github at octeontx-kmod along with build, install and dpdk usage instructions.

Note: The PF driver and the required microcode for the crypto offload block will be available with OCTEON TX SDK only. So for using crypto offload, follow the steps mentioned in *Setup Platform Using OCTEON TX SDK*.

2. **ARM64 Tool Chain**

   For example, the *aarch64* Linaro Toolchain, which can be obtained from here.

3. **Rootfile system**

   Any *aarch64* supporting filesystem can be used. For example, Ubuntu 15.10 (Wily) or 16.04 LTS (Xenial) userland which can be obtained from http://cdimage.ubuntu.com/ubuntu-base/releases/16.04/release/ubuntu-base-16.04.1-base-arm64.tar.gz.

As an alternative method, Platform drivers can also be executed using images provided as part of SDK from Cavium. The SDK includes all the above prerequisites necessary to bring up a OCTEON TX board. Please refer *Setup Platform Using OCTEON TX SDK*.

- Follow the DPDK `../linux_gsg/index` to setup the basic DPDK environment.

4.3 Setup Platform Using OCTEON TX SDK

The OCTEON TX platform drivers can be compiled either natively on **OCTEON TX** ® board or cross-compiled on an x86 based platform.

The **OCTEON TX** ® board must be running the linux kernel based on OCTEON TX SDK 6.2.0 patch 3. In this, the PF drivers for all hardware offload blocks are already built in.

4.3.1 Native Compilation

If the kernel and modules are cross-compiled and copied to the target board, some intermediate binaries required for native build would be missing on the target board. To make sure all the required binaries are available in the native architecture, the linux sources need to be compiled once natively.

```
cd /lib/modules/$(uname -r)/source
make menuconfig
make
```

The above steps would rebuild the modules and the required intermediate binaries. Once the target is ready for native compilation, the OCTEON TX platform drivers can be compiled with the following steps,

```
cd <dpdk directory>
make config T=arm64-thunderx-linux-gcc
make
```

The example applications can be compiled using the following:

```
cd <dpdk directory>
export RTE_SDK=$PWD
export RTE_TARGET=build
cd examples/<application>
make
```

4.3.2 Cross Compilation

The DPDK applications can be cross-compiled on any x86 based platform. The OCTEON TX SDK need to be installed on the build system. The SDK package will provide the required toolchain etc.

Refer to `../linux_gsg/cross_build_dpdk_for_arm64` for further steps on compilation. The ‘host’ & ‘CC’ to be used in the commands would change, in addition to the paths to which libnuma related files have to be copied.

The following steps can be used to perform cross-compilation with OCTEON TX SDK 6.2.0 patch 3:

```
cd <sdk_install_dir>
source env-setup

git clone https://github.com/numactl/numactl.git
cd numactl
git checkout v2.0.11 -b v2.0.11
./autogen.sh
autoconf -i
./configure --host=aarch64-thunderx-linux CC=aarch64-thunderx-linux-gnu-gcc --prefix=<numa inst
make install
```

The above steps will prepare build system with numa additions. Now this build system can be used to build applications for **OCTEON TX** ® platforms.

```
cd <dpdk directory>
export RTE_SDK=$PWD
export RTE_KERNELDIR=$THUNDER_ROOT/linux/kernel/linux
make config T=arm64-thunderx-linux-gcc
make -j CROSS=aarch64-thunderx-linux-gnu- CONFIG_RTE_KNI_KMOD=n CONFIG_RTE_EAL_IGB_UIO=n EXTRA_
```

If NUMA support is not required, it can be disabled as explained in `../linux_gsg/cross_build_dpdk_for_arm64`.

Following steps could be used in that case.

```
make config T=arm64-thunderx-linux-gcc
make CROSS=aarch64-thunderx-linux-gnu-
```

SDK and related information can be obtained from: Cavium support site.