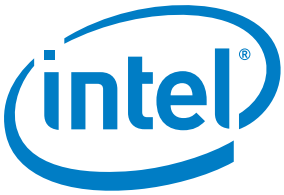


FlexRAN Reference Solution NB-IoT

User Guide

April 2018

Intel Confidential

Document Number: 575823-1.1

Contents

Figures

Tables

Revision History

Document Number	Revision	Description	Date
575823	1.1	Correct typographical, description, and formatting errors	April 2018
575823	1.0	Initial release	January 2018

§

1 Introduction

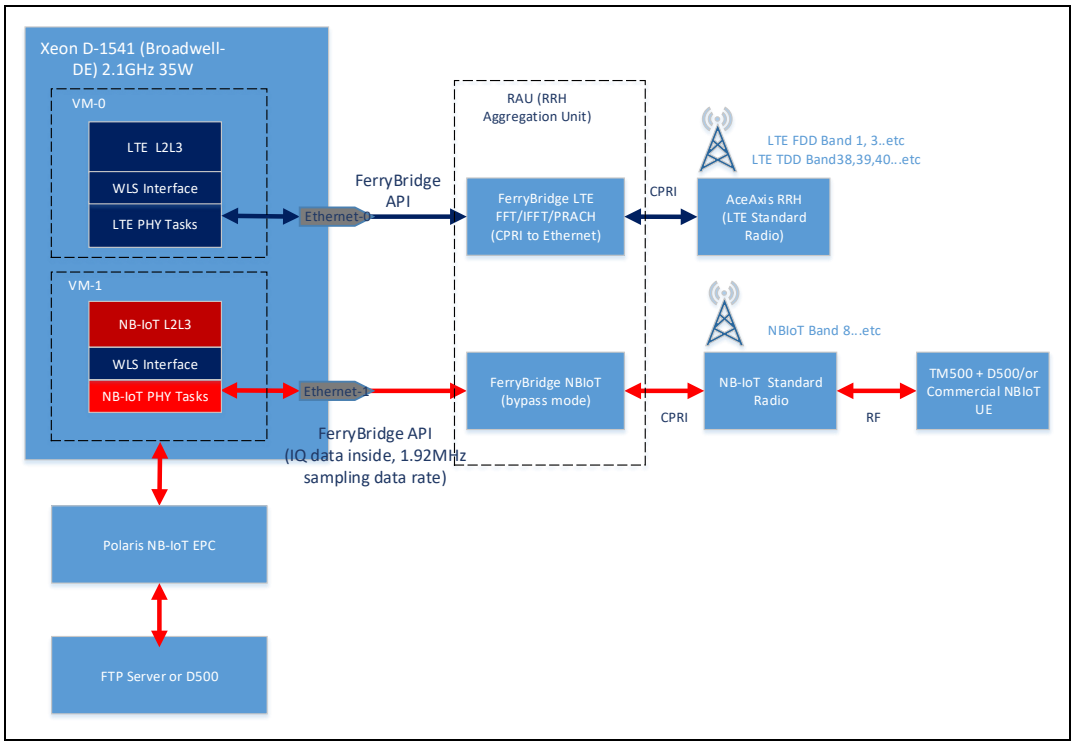
The *FlexRAN NB-IoT eNB L User Guide* describes the overall setup and components required to demonstrate the capabilities and performance of a FlexRAN NB-IoT implementation based on an Intel® Xeon® processor using Wind River* Open Virtualization Platform 6 (OVP6).

FlexRAN stands for a Flexible and Programmable Platform for Software-Defined Radio Access Networks.

The goal of this document is to enable users to quickly configure, build, and run FlexRAN NB-IoT L1 applications on an Intel® Xeon® processor-based platform. The NB-IoT L1 application runs in standalone mode and uses a different binary than FlexRAN LTE.

1.1 Overview

Figure 1. FlexRAN NB-IoT Block Diagram (Co-deployment with FlexRAN LTE)



The FlexRAN hardware platform consists of a MaxCore* platform with an Artesyn* SharpServer* PCIe*-7410 card with dual Intel® Xeon® D-1541 processors running at 2.1 GHz.

One central processing unit (CPU) or virtual machine (VM) can be used for one cell of 20M bandwidth and two carriers of 3GPP release 10. The MaxCore* chassis is configured to map a PCIe* slot with an Intel® 82599ES 10 GB Ethernet controller to an Intel® Xeon® D CPU, which in turn is connected via a 10 GB optical Ethernet to a Ferry Bridge field programmable gate array (FPGA) that converts Ethernet protocols to the Common Public Radio Interface (CPRI). The Ferry Bridge FPGA's CPRI port then connects to an AceAxis* Advanced Radio Tester (ART) remote radio head (RRH). Radio Frequency (RF) ports of the RRH are connected to either commercial user equipment (CUEs) or a Cobham* TM500 multi-UE emulator. The Artesyn* blade runs the eNodeB function of a long-term evolution (LTE) network. The setup includes an Evolved Packet Core (EPC) and a Linux* FTP server/Cobham* D500 to perform end-to-end testing data of traffic over the LTE network.

Note: For more details, refer to the *FlexRAN LTE User Guide* (refer to Table 2).

Another CPU (or VM) can be used for FlexRAN NB-IoT, as described in this document. The FlexRAN NB-IoT L1 application runs in standalone mode, with a peak Unacknowledged Mode DL throughput of 68 Kbps, and all IFFT/FFT/PRACH calculations are done by the NB-IoT L1 application. The Ferry Bridge NB-IoT is running in bypass mode, and receiving/sending NB-IoT time domain IQ sampling data (sampling rate is 1.92 million (M) samples per second). The Intel Ferry Bridge FPGA module does not support the NB-IoT sampling rate, so customers need to update their own FGPA module to support NB-IoT.

For NB-IoT standard radio, contact the vendor for NB-IoT band radio (for standalone mode, NB-IoT radio band is the same for the GSM band). The proposed setup also includes a Polaris Networks* NB-IoT EPC and a Cobham* TM500 (support NB-IoT).

1.2 Hardware

The hardware configuration used for NB-IoT testing:

- Artesyn* SharpServer* PCIe-7410 with two 8-core Intel® Xeon® processors at 2.1 GHz
- Intel® 82599 10 Gigabit Ethernet Controller (NIC)
- Ferry Bridge FPGA module (customers update their own FPGA modules)
- RRH—off-the-shelf RRH (needs to support the NB-IoT standalone radio band and 1.92M sampling rate)
- FTP server
- Cobham* D500 traffic generation
- User equipment (UE) side:
  - Commercial standalone NB-IoT user equipment (UE) module
  - Cobham* TM500 Multi-UE emulator (standalone NB-IoT)
- 1-Gigabit switch

1.3 Acronyms

Table 1. Acronyms

Term	Description
ART	Advanced Radio Tester
BBU	Base Band Unit
BSP	Board Support Package
CPRI	Common Public Radio Interface
CPU	Central Processing Unit
CUE	Commercial User Equipment
DPDK	Data Plane Development Kit
EPC	Evolved Packet Core
FlexRAN	Flexible and Programmable Platform for Software-Defined Radio Access Networks
FPGA	Field Programmable Gate Array
FTP	File Transfer Protocol
L1	Layer 1 or Physical Layer
L2	Layer 2 or Media Access Control Layer
LTE	Long-Term Evolution (a 4G mobile networking standard)
NB-IoT	Narrow Band—Internet of Things
NIC	Network Interface Card
OTA	Over the Air
OVP6	Open Virtualization Platform 6
PCIe*	Peripheral Component Interconnect Express
PF	Physical Function
RAN	Radio Access Network
RF	Radio Frequency
RRH	Remote Radio Head (such as AceAxis* ART)
SISO	Single Input, Single Output
UE	User Equipment

Term	Description
VF	Virtual Function
VM	Virtual Machine
WR	Wind River*

1.4 Reference Documents and Resources

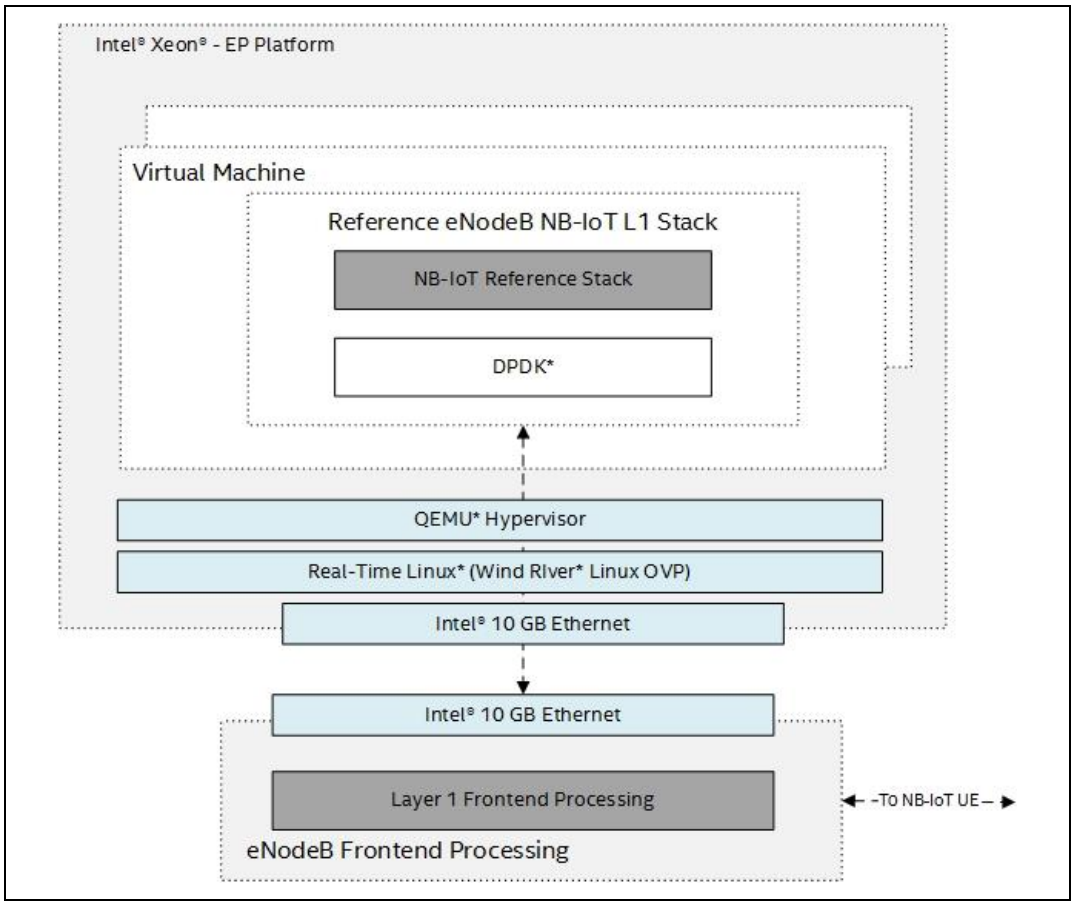
Table 2. Reference Documents and Resources

Title	Document Number/ Location
FlexRAN Reference Solution L1 User Guide	570228
Intel® FlexRAN Board Support Package (BSP) Release Notes	571739
FlexRAN Reference Solution L1 XML Configuration User Guide	571741
FlexRAN Reference Solution Software v1.5.0 Release Notes	575822
FlexRAN Reference Solution NB-IOT L2-L1 API Specification	575824
MaxCore Platform*	https://www.artesyn.com/computing/products/product/max-core
Advanced Radio Tester Product Fact Sheet	http://aceaxis.co.uk/website/wp-content/uploads/2016/07/AdvancedRadioTester_July2016.pdf
TM500 LTE TEST Mobile Application User Guide	https://www.aeroflex.com
Wind River Linux* Open Virtualization Profile, Virtual Node User Guide, 6.0*	www.windriver.com
Wind River Linux User Guide, 6.0*	www.windriver.com
Intel® C++ Compiler in Intel® Parallel Studio XE	https://software.intel.com/en-us/c-compilers/ipsxe
DPDK Documentation	http://dpdk.org/doc/guides
DPDK 17..11	http://dpdk.org/browse/dpdk/snapshot/dpdk-17.11.tar.gz
USRP Hardware Driver and USRP Manual	http://files.ettus.com/manual/page_install.html

1.5 FlexRAN Application Environment

The software stack for FlexRAN applications is based on Wind River* OVP6 for both host and guest OS. The user is expected to build and install Wind River* OVP6 in the build environment before going further in this document. Figure 2 shows the FlexRAN software stack.

Figure 2. FlexRAN Software Stack



The following versions of OVP6 have been tested with the current release of NB-IoT eNodeB:

- Host OS Version: Wind River* OVP 6.0.0.25
- Guest OS Version: Wind River* OVP 6.0.0.23

In general, other versions of Wind River* OVP can be used as well, because no direct dependency occurs between NB-IoT eNodeB applications and OVP components. However, Intel recommends using a proven combination of versions to avoid unexpected issues with the build and configuration of the FlexRAN application.

§

2 WR-OVP6 SDK Compilation

This section describes the steps used to generate Wind River* OVP6 host, guest, and installer images used with the FlexRAN application. Wind River* documentation provides additional information, including usage, on the following commands.

The default configuration of the PCIe switch on the Artesyn* MaxCore* chassis does not support enabling the CONFIG_PCIEASPM option for PCIe Power Management. Use the Linux* kernel boot option `pcie_aspm=off` for host and guest to disable PCIe power management.

Appendix A provides kernel configuration changes versus default OVP6.

2.1 Guest (VM) Image

This section provides code to prepare, configure, and compile the guest (VM) image.

2.1.1 Prepare and Configure

```
#!/bin/sh

PROJECT_HOME=`pwd`
export WIND_HOME=/home/user/wr-ovp6 #location where wr-ovp6 is
installed.

cd $WIND_HOME
mkdir -p sscache
export SSTATE_DIR=$WIND_HOME/sscache

mkdir -p ccache
export CCACHE_DIR=$WIND_HOME/ccache

export BSP=x86-64-kvm-guest
export ROOTFS=ovp-guest+initramfs-integrated

cd $PROJECT_HOME
export PROJECT=$PROJECT_HOME/ovp-guest
mkdir -p $PROJECT
cd $PROJECT

$WIND_HOME/wrlinux-6/wrlinux/configure \
--with-rcpl-version=0023 \
--enable-board=$BSP \
--enable-rootfs=$ROOTFS \
```

```
--enable-parallel-pkgbuilds=6 \
--enable-jobs=6 \
--enable-ccache=yes \
--with-ccache-dir=$WIND_HOME/ccache \
--with-sstate-dir=$WIND_HOME/sstate \
--enable-reconfig \
--enable-addons=wr-ovp \
--with-template=feature/debug,feature/gdb,feature/target-
toolchain
```

2.1.2 Compile

```
# make
```

2.2 Host Image

This section provides information on how to prepare, configure, and compile the host image.

2.2.1 Prepare and Configure

```
#!/bin/sh

PROJECT_HOME=`pwd`
export WIND_HOME=/home/user/wr-ovp6  #location where wr-ovp6 is
installed

cd $WIND_HOME
mkdir -p sscache
export SSTATE_DIR=$WIND_HOME/sscache
mkdir -p ccache
export CCACHE_DIR=$WIND_HOME/ccache

export BSP=intel-x86-64
export ROOTFS=ovp-kvm+installer-support

cd $PROJECT_HOME
export PROJECT=$PROJECT_HOME/ovirt-node-ovp-guest
mkdir -p $PROJECT
cd $PROJECT
```

```
#configure ovp-host
$WIND_HOME/wrlinux-6/wrlinux/configure \
--with-rcpl-version=0025 \
--enable-board=$BSP \
--enable-rootfs=$ROOTFS \
--enable-addons=wr-ovp \
--with-layer=wr-kvm-binary-guest-images \
--with-kvm-guest-kernel=$PROJECT_HOME/ovp-
guest/export/images/bzImage-x86-64-kvm-guest.bin \
--with-kvm-guest-img=$PROJECT_HOME/ovp-
guest/export/images/wrlinux-image-initramfs-x86-64-kvm-
guest.cpio.gz \
--enable-parallel-pkgbuilds=6 \
--enable-jobs=6 \
--enable-ccache=yes \
--with-ccache-dir=$WIND_HOME/ccache \
--with-sstate-dir=$WIND_HOME/sstate \
--enable-reconfig \
--enable-internet-download=yes \
--with-template=feature/debug,feature/gdb,feature/self-
hosted,feature/target-toolchain,feature/kernel-
tune,feature/libhugetlbfs,feature/analysis,feature/system-stats
```

2.2.2 Compile

```
# make
```

2.3 Installer Image

The installer is a small USB bootable image, which boots from a USB on the target machine and installs the virtio-node (host) image on the target machine hard-drive. The host image includes the guest (VM) image at `/opt/ location`.

2.3.1 Prepare and Configure

```
#!/bin/sh

PROJECT_HOME=`pwd`
export WIND_HOME=/home/user/wr-ovp6  #location where wr-ovp6 sdk
is installed

cd $WIND_HOME
mkdir -p sscache
export SSTATE_DIR=$WIND_HOME/sscache
```

```
mkdir -p ccache
export CCACHE_DIR=$WIND_HOME/ccache

export BSP=intel-x86-64
export KERNEL=standard
export ROOTFS=wr-installer

cd $PROJECT_HOME
export PROJECT=$PROJECT_HOME/ovirt-node-ovp-guest-installer
mkdir -p $PROJECT
cd $PROJECT

$WIND_HOME/wrlinux-6/wrlinux/configure \
--with-rcpl-version=0023 \
--enable-board=$BSP \
--enable-kernel=$KERNEL \
--enable-rootfs=$ROOTFS \
--enable-target-installer=yes \
--enable-bootimage=iso \
--with-installer-target-build=$PROJECT_HOME/ovirt-node-ovp-
guest/export/images/wrlinux-image-ovp-kvm-intel-x86-64.ext3 \
--enable-parallel-pkgbuilds=6 \
--enable-jobs=6 \
--enable-ccache=yes \
--with-ccache-dir=$WIND_HOME/ccache \
--with-sstate-dir=$WIND_HOME/sstate \
--enable-reconfig
```

2.3.2 Compile

```
# make

# make usb-image
```

After successful compilation, find the installer .iso image at `$PROJECT_HOME/ovirt-node-ovp-guest-installer/export/*.iso`.

§

3 Target Platform Configuration

3.1 WR-OVP6 Installation

For the Artesyn* Blade console configuration, refer to the MaxCore* platform documentation (refer to Table 2). Each blade has a UART-to-USB port connected to both CPUs on the blade. The user needs to modify the original Wind River* installation USB image to support the correct UART speed, as follows:

```
EFI/BOOT/grub.cfg

# Automatically created by OE
#serial --unit=0 --speed=115200 --word=8 --parity=no --stop=1
serial --speed 38400 --unit=0 --word=8 --parity=no --stop=1
default=boot
timeout=10

menuentry 'boot'{
#linux /vmlinuz LABEL=boot root=/dev/ram0
linux /vmlinuz LABEL=boot console=ttyS0,38400 root=/dev/ram0
initrd /initrd
}
```

Use the USB installer image to install the host image on the target hardware. Installation of Wind River* images using the installer is similar to a typical Linux* distribution installation process. The *Wind River Linux User Guide, 6.0* (refer to Table 2) provides additional information on how to install WR Linux using the installer image.

3.2 Software Configuration

This section describes how to configure the host, guest, and network.

3.2.1 Host Configuration

This section provides the host BIOS and OS configuration. It also provides information on how to log on to the host.

3.2.1.1 Recommended BIOS Configuration

The recommended BIOS configuration is as follows:

- Advanced → Processor Configuration
  - Enhanced Intel SpeedStep® Technology **Disabled**
  - Processor C3 **Disabled**
  - Processor C6 **Disabled**
  - Intel Hyper-Threading Tech **Disabled**
  - Active Processor Cores **ALL**
  - Intel Virtualization Technology **Enabled**
  - Intel VT for Directed I/O **Enabled**
  - Coherency Support **Disabled**
- Advanced →Memory Configuration
  - Memory Power Optimization **Performance Optimized**
- Advanced →PCI configuration → NIC Controller
  - Wake-on-LAN (PME) **Disabled**
  - NIC1 Port 1 PXE **Disabled**
  - NIC1 Port2 PXE **Disabled**
- Advanced →USB Configuration
  - Legacy USB Support **Auto**
  - Port 60/64 Emulation **Enabled**
  - Make USB Devices Non-Bootable **Disabled**
- Server Management
  - Assert NMI on SERR **Disabled**

3.2.1.2 Recommended Kernel Command Line Arguments for Host

```
<Host>#cat /proc/cmdline
BOOT_IMAGE=/vmlinuz-3.10.91-ovp-rt97-WR6.0.0.25_preempt-rt
root=/dev/sda2 rw console=ttyS0,38400 clock=pit rcu_nocbs=1-7
rcu_nocb_poll=1 isolcpus=1-7 irqaffinity=0 tsc=perfect selinux=0
enforcing=0 intel_iommu=on iommu=pt default_hugepagesz=1G
hugepagesz=1G hugepages=21 nohz_full=1-7 intel_pstate=disable
idle=poll noswap pci=pcie_bus_perf
```

3.2.1.3 Host Logon Details

The user and password for host logon are:

- User: root
- Password: root123

3.2.2 Guest Configuration

This section provides information on how to configure the guest for the kernel and how to log on to the guest.

3.2.2.1 Recommended Kernel Command Line Arguments for Guest

```
<Guest># cat /proc/cmdline
root=/dev/vda ro console=ttyS0 isolcpus=1-6 irqaffinity=0
nohz_full=0-6 clocksource=tsc tsc=perfect selinux=0 enforcing=0
default_hugepagesz=1G hugepagesz=1G hugepages=10 noswap
pci=pcie_bus_perf
```

3.2.2.2 Guest Logon Details

The user and password for guest logon are:

- User: root
- Password: root

3.2.3 Networking Configuration

By default, OVP6 networking is configured to use bridging between host and guest systems. No other networking interfaces are configured. For a FlexRAN eNodeB application running in a virtual machine, the following changes are required. In addition, the Artesyn* MaxCore* Platform must be configured to map particular Ethernet ports (virtual or physical) to the given CPU used for the FlexRAN application.

3.2.3.1 Assignment of Ethernet Ports to Intel® Xeon® Processor D-xxxx Product Family CPU

For detailed information on the PCIe switch functionality and the Ethernet ports available in the system, refer to the *Artesyn* MaxCore* User Guide* (refer to Table 2). This document provides a minimal set of changes to run one cell with the NB-IoT eNodeB application.

Overall, the Intel® Xeon® processor running the eNodeB application requires two virtual Ethernet (VE) ports—one for the host system to perform control and management of VMs, and one for the guest S1/eGTP connection to the Evolved Packet Core (EPC).

For an RRH connection via Ferry Bridge, the dual port Intel® 82599ES 10 Gigabit Ethernet controller has to be mapped to the CPU and configured in pass-through mode to be used by the virtual machine.

From the master CPU on the MaxCore* Platform (slot 1 CPU 1), the following commands need to be executed to correctly set up networking for the FlexRAN application. Any given port number and slot number are provided *as an example*, and users are expected to modify the settings according to the physical configuration of the chassis being used. For more information, refer to the *FlexRAN Reference Solution L1 User Guide* (refer to Table 2).

```
#List current assignment
mccs_tool.py --method=list-assigned-funcs

#assign 2 additional virtual functions from slot 16 ETH to slot 1
cpu 2
mccs_tool.py --method=assign-func --cpu=1,2,1 --func=16,1,6
mccs_tool.py --method=assign-func --cpu=1,2,1 --func=16,1,7

#assign Intel® 82599ES 10 Gigabit Ethernet Controller connected
to slot 7 to slot 1 cpu mccs_tool.py --method=assign-func --
cpu=1,2,1 --func=7,1,1
mccs_tool.py --method=assign-func --cpu=1,2,1 --func=7,2,1

#apply updated settings
mccs_tool.py --method=apply-config
```

After performing the commands, the system must be power cycled. The new settings are applied on the next boot of the master CPU (slot 1 CPU 1).

3.2.3.2 Configuring Ethernet Networking

The MaxCore* platform has two switches and an additional slot 16 with network control that is used to access the external 1 GB Ethernet network.

The given configuration uses the slot 16 device as a port physically connected to the external network (where the EPC is connected). Describing a configuration where the EPC runs on the same MaxCore* chassis is outside the scope of this document.

The following example shows a configuration of the host with DHCP:

```
root@art-pcie7410-s5-c2:/opt/vm# lspci |grep Eth
04:00.0 Ethernet controller: Intel Corporation Device 15ab
04:00.1 Ethernet controller: Intel Corporation Device 15ab
08:11.0 Ethernet controller: Intel Corporation 82599 Ethernet
Controller Virtual Function (rev 01)
08:11.2 Ethernet controller: Intel Corporation 82599 Ethernet
Controller Virtual Function (rev 01)
09:00.0 Ethernet controller: Intel Corporation 82599ES 10-Gigabit
SFI/SFP+ Network Connection (rev 01)
```

```
09:00.1 Ethernet controller: Intel Corporation 82599ES 10-Gigabit
SFI/SFP+ Network Connection (rev 01)

ethtool -i eth0
driver: ixgbevf
version: 2.7.12-k
firmware-version:
bus-info: 0000:08:11.0
supports-statistics: yes
supports-test: yes
supports-eeprom-access: no
supports-register-dump: yes
supports-priv-flags: no

cat /etc/network/interfaces
# The loopback interface
auto lo
iface lo inet loopback

# Wireless interfaces
iface wlan0 inet dhcp
        wireless_mode managed
        wireless_essid any
        wpa-driver wext
        wpa-conf /etc/wpa_supplicant.conf

iface atml0 inet dhcp

# Wired or wireless interfaces
auto eth0
iface eth0 inet dhcp
```

For this example, Ethernet ports assigned to VM are:

```
08:11.2 Ethernet controller: Intel Corporation 82599 Ethernet
Controller Virtual Function (rev 01)

09:00.0 Ethernet controller: Intel Corporation 82599ES 10-Gigabit
SFI/SFP+ Network Connection (rev 01)
```

Where `08:11.2` is the control plane and data plane of the eNodeB application, and `09:00.0` is the 10-Gigabit interface for RRH connection. Corresponding ports mapped to VM can be configured via QEMU* start-up script, as follows:

```
#cat /opt/vm/q_vm1_xd.sh
…
export VM_HOME=`pwd`
export AFFINITY_ARGS="taskset -c 1,2,3,4,5,6,7"
export QEMU=/usr/bin/qemu-system-x86_64
export DPDK_ARGS="-c 0x0f -n 4 --proc-type=secondary -- -enable-
dpdk"

/opt/dpdk-16.11/tools/dpdk_nic_bind.py --unbind 0000:08:11.2
/opt/dpdk-16.11/tools/dpdk_nic_bind.py --unbind 0000:09:00.0
/opt/dpdk-16.11/tools/dpdk_nic_bind.py --unbind 0000:09:00.1

$AFFINITY_ARGS $QEMU\
    -cpu host \
    -nographic \
    -k en-us \
    -m 20480 \
    -mem-path /mnt/huge \
    -mem-prealloc \
    -mlock \
    -name GUEST_1 \
    -enable-kvm \
    -smp cpus=7,cores=7,threads=1,sockets=1 \
    -vcpu 0,affinity=0x0002,prio=0 \
    -vcpu 1,affinity=0x0004,prio=95 \
    -vcpu 2,affinity=0x0008,prio=95 \
    -vcpu 3,affinity=0x0010,prio=95 \
    -vcpu 4,affinity=0x0020,prio=95 \
    -vcpu 5,affinity=0x0040,prio=95 \
    -vcpu 6,affinity=0x0080,prio=95 \
    -kernel $VM_HOME/vm1/bzImage_3.10.91-ovp-rt97-
WR6.0.0.25_preempt-rt \
    -append 'root=/dev/vda ro console=ttyS0 isolcpus=1-6
irqaffinity=0 nohz_full=0-6 clocksource=tsc tsc=perfect selinux=0
enforcing=0 default_hugepagesz=1G hugepagesz=1G hugepages=10
noswap idle=poll' \
    -drive file=$VM_HOME/vm1/wrlinux-image-ovp-kvm-x86-64-kvm-
guest.ext3,if=virtio \
    -drive file=$VM_HOME/vm1/disk_ph3.qcow2,if=none,id=drive-ide0-
0-0,format=qcow2 \
    -device ide-hd,bus=ide.0,unit=0,drive=drive-ide0-0-0,id=ide0-
0-0,bootindex=1 \
    -device pci-assign,host=0000:09:00.0,vcpuaffine \
    -device pci-assign,host=0000:09:00.1,vcpuaffine \
    -device pci-assign,host=0000:08:11.2,vcpuaffine
```

After the VM is booted, the corresponding configuration of the Ethernet port has to be done for guest, as shown:

```
tm500-xeon-d-vm0# lspci |grep
00:03.0 Ethernet controller: Intel Corporation 82540EM Gigabit
Ethernet Controller (rev 03)
00:04.0 Ethernet controller: Intel Corporation 82599ES 10-Gigabit
SFI/SFP+ Network Connection (rev 01)
00:05.0 Ethernet controller: Intel Corporation 82599ES 10-Gigabit
SFI/SFP+ Network Connection (rev 01)
00:06.0 Ethernet controller: Intel Corporation 82599 Ethernet
Controller Virtual Function (rev 01)

tm500-xeon-d-vm0:# ethtool –i eth1
driver: ixgbevf
version: 2.7.12-k
firmware-version:
bus-info: 0000:00:06.0
supports-statistics: yes
supports-test: yes
supports-eeprom-access: no
supports-register-dump: yes
supports-priv-flags: no

tm500-xeon-d-vm0:#cat /etc/network/interfaces
# /etc/network/interfaces -- configuration file for ifup(8),
ifdown(8)

# The loopback interface
auto lo
iface lo inet loopback

# Wireless interfaces
iface wlan0 inet dhcp
        wireless_mode managed
        wireless_essid any
        wpa-driver wext
        wpa-conf /etc/wpa_supplicant.conf

iface atml0 inet dhcp

# Wired or wireless interfaces
auto eth1
iface eth1 inet dhcp
```

```
tm500-xeon-d-vm0:#ifconfig
eth1      Link encap:Ethernet  HWaddr 02:01:00:10:01:07
          inet addr:10.233.183.18  Bcast:10.233.183.255
Mask:255.255.252.0
          inet6 addr: fe80::1:ff:fe10:107/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2883 errors:0 dropped:1 overruns:0 frame:0
          TX packets:59 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:380166 (371.2 KiB)  TX bytes:10244 (10.0 KiB)

#corresponding 10 Gb port has to be provided to L1 application
via dpdk_vm.sh
cat ./dpdk.sh
#! /bin/bash
...
$RTE_SDK/tools/dpdk_nic_bind.py --bind=igb_uio 0000:00:04.0
```

3.3 Intel® C++ Compiler

The Intel® C++ Compiler is required to compile Intel Data Plane Development Kit (DPDK) and L1 software. The recommended Intel® C++ Compiler version is 17.0.1. Instructions on where to obtain and how to install the compiler package are outside the scope of this document. The Intel DPDK and L1 packages must be compiled with the Intel® C++ Compiler.

3.4 Intel Data Plane Development Kit (DPDK) Configuration

The FlexRAN application requires Intel DPDK version 17.11. To download and configure the DPDK, follow these steps:

1. Download DPDK 17.11 at http://dpdk.org/browse/dpdk/snapshot/dpdk-17.11.tar.gz (also included in Table 2).
2. Unzip to `/opt/dpdk-17.11`.
3. Apply the changes (patch) to DPDK, which add changes to support Wind River* OVP6 and IRQ handling in the user space:

```
/opt/dpdk-17.11 # patch -p1 < /<path-to-patch-file>/dpdk-
17.11.patch
```

4. Configure DPDK:

```
/opt/dpdk-17.11# cd ./tools/
/opt/dpdk-17.11/tools# ./dpdk-setup.sh

Select
[14] x86_64-native-linuxapp-icc
Select
[16] Insert IGB UIO module

Exit from dpdk-setup.sh
```

The following shows the DPDK patch to compile 17.11 with OVP6 and IRQ mode in user space:

```
diff --git a/drivers/net/ixgbe/ixgbe_ethdev.c
b/drivers/net/ixgbe/ixgbe_ethdev.c
old mode 100644
new mode 100755
index edc9b22..f6cc778
--- a/drivers/net/ixgbe/ixgbe_ethdev.c
+++ b/drivers/net/ixgbe/ixgbe_ethdev.c
@@ -88,7 +88,7 @@
 #define IXGBE_FC_LO    0x40

 /* Default minimum inter-interrupt interval for EITR
configuration */
-#define IXGBE_MIN_INTER_INTERRUPT_INTERVAL_DEFAULT    0x79E
+#define IXGBE_MIN_INTER_INTERRUPT_INTERVAL_DEFAULT   0
//(0x79E/8)

 /* Timer value included in XOFF frames. */
 #define IXGBE_FC_PAUSE 0x680
@@ -2241,7 +2241,8 @@ struct rte_ixgbe_xstats_name_off {
          if (intr_vector > IXGBE_MAX_INTR_QUEUE_NUM) {
                  PMD_INIT_LOG(ERR, "At most %d intr queues
supported",

    IXGBE_MAX_INTR_QUEUE_NUM);
-                 return -ENOTSUP;
+//                 return -ENOTSUP;
+            intr_vector = IXGBE_MAX_INTR_QUEUE_NUM;
          }
          if (rte_intr_efd_enable(intr_handle, intr_vector))
                  return -1;
```

```
diff --git a/lib/librte_eal/linuxapp/kni/ethtool/igb/kcompat.h
b/lib/librte_eal/linuxapp/kni/ethtool/igb/kcompat.h
old mode 100644
new mode 100755
index 84826b2..0c185e3
--- a/lib/librte_eal/linuxapp/kni/ethtool/igb/kcompat.h
+++ b/lib/librte_eal/linuxapp/kni/ethtool/igb/kcompat.h
@@ -3870,11 +3870,13 @@ static inline struct sk_buff
*__kc__vlan_hwaccel_put_tag(struct sk_buff *skb,
 #if (!(SLE_VERSION_CODE == SLE_VERSION(12,0,0)))
 #ifdef NETIF_F_RXHASH
 #define PKT_HASH_TYPE_L3 0
+#if 0
 static inline void
 skb_set_hash(struct sk_buff *skb, __u32 hash, __always_unused
int type)
 {
    skb->rxhash = hash;
 }
+#endif
 #endif /* NETIF_F_RXHASH */
 #endif /* < SLES12 */
 #endif /* < 3.13.0-30.54 (Ubuntu 14.04) */
diff --git a/lib/librte_ether/rte_ethdev.c
b/lib/librte_ether/rte_ethdev.c
old mode 100644
new mode 100755
```

§

4 FlexRAN NB-IoT Applications

4.1 Prerequisites for Compilation

The prerequisites for compilation are:

- The virtual machine is required to be built with the GCC toolchain installed on the target. See Section 4.5.
- An Intel compiler must be installed in the virtual machine.
- The kernel source files must be available for the compilation of the `wls.ko` module. Appendix A provides information regarding the kernel source files included in the virtual machine. By default, the OVP6 build does not include kernel sources.
- The FlexRAN NB-IoT software package is comprised of:
  - Ferry Bridge library source code
  - wls_mod source code
  - NB-IoT PHY source code:
    - Standalone NB-IoT
    - 3.75K subcarrier spacing
    - SISO
- NB-IoT Layer 2/3 (L2L3 vendor-provided solution)
- CMake

The following commands are used for downloading, building, and installing CMAKE:

```
wget https://cmake.org/files/v3.9/cmake-3.9.2.tar.gz --no-check-certificate
tar xvzf cmake-3.9.2.tar.gz
cd cmake-3.9.2
./configure
make
sudo make install
```

4.2 NB-IoT L1 Application

The NB-IoT L1 application is a separate binary from the LTE L1 application, and it runs in standalone mode. Some source code folders for NB-IoT are separate from those of LTE L1, but for some common functionalities, it shares folders with LTE L1: SDK, BBUPoolingFramework, WLS, PHY Ferry Bridge API, PHY Utils, and so forth.

To extract all components of the FlexRAN release, use `extract.sh`. For example:

```
/home/turner/work/flexran
├── bin/lte/l1/lte_phy_nbiot
├── build/lte/l1app_nbiot
├── doxygen
├── ferrybridge
├── framework
├── misc
├── sdk
├── source/lte/threads_nbiot
├── source/lte/fec_dec_nbiot
├── source/lte/api_nbiot
├── source/lte/dl_mod_nbiot
├── source/lte/ul_demod_nbiot
├── wls_libs
└── wls_mod
```

To build the L1 application, follow these steps:

1. Set up the environment:

```
source /opt/intel/bin/iccvars.sh  intel64 -platform linux
export RTE_SDK=/opt/dpdk-17.11
export DIR_WIRELESS_FW=/home/turner/work/flexran/framework
export DIR_WIRELESS_SDK_ROOT=/home/turner/work/flexran/sdk
```

For AVX2 system (such as Broadwell):

```
export SDK_BUILD=build-avx2-icc
export WIRELESS_SDK_TARGET_ISA=avx2
export DIR_WIRELESS_SDK=${DIR_WIRELESS_SDK_ROOT}/${SDK_BUILD}
```

For AVX512 system (such as Skylake):

```
export SDK_BUILD=build-avx512-icc
export WIRELESS_SDK_TARGET_ISA=avx512
export DIR_WIRELESS_SDK=${DIR_WIRELESS_SDK_ROOT}/${SDK_BUILD}
```

2. For a first-time installation, build the Ferry Bridge library, as follows:

```
cd ./ferrybridge/lib
make clean;
make;
```

   After this step is completed, in a new setup, there is no need to rebuild the Ferry Bridge library in order to build the L1 FlexRAN application.

3. For a first-time installation, build the `mlog` library, as follows:

```
cd ./wls_lib/mlog
./build.sh
```

   After this step is completed, in a new setup, there is no need to rebuild the `mlog` library.

4. For a first-time installation, build the `wls` interface module and library, as follows:

```
cd ./wls_mod/
./build.sh clean
./build.sh
```

   After this step is completed for a new setup, there is no need to rebuild `libwls.so` and `wls.ko`.

5. Build the wireless SDK component. The SDK supports compilation for either the AVX2 or AVX512 system, depending on your environment variables, as per Step 1:

```
cd /home/turner/work/flexran/sdk/
./create-makefiles-linux.sh
```

   For AVX2 systems:

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

   For AVX512 systems:

```
cd build-avx512-icc
make install
```

6. Build the Framework component:

```
cd /home/turner/work/framework/bbupool
make clean
make
```

7. Build the L1 application:

Note: The default compilation with `BBU_POOL=1` does not support compilation with `FEC_HW_ACCEL` enabled.

```
cd ../build/l1app_nbiot
./build.sh xclean
./build.sh
/build/l1app_nbiot$ ./build.sh
Number of commandline: 0
Build using xHost params
RELEASEBUILD=1
BUILD_OPT: Host
DEV_DETECT: NON_AVX512
CPU_TYPE: 0
COMMAND_LINE=
=====================================================
Building l1app:
     BUILD_OPT = Host
     RTE_TARGET = x86_64-native-linuxapp-icc
     DEVICE = NON_AVX512
     RTE_SDK = /opt/dpdk-17.11
     DIR_WIRELESS_SDK = /home/turner/work/flexran/sdk/build-
avx2-icc
     DIR_WIRELESS_FW = /home/turner/work/flexran/sdk/framework
     WCK_DIR =
     BUILD = Release
     BBU_POOL = 1
     FEC_HW_ACCEL = 0
=====================================================
=====================================================
[BUILD] lib : physrc
[AR] libphysrc_r.a
=====================================================
[BUILD] lib : auxlib
[AR] libauxlib_r.a
=====================================================
```

```
[BUILD] elf : l1app
[CC] threads/main.o
...
```

8. Find the resulting NB-IoT PHY binary here:

```
  ./lte_phy/lte_phy_nbiot
|-- dpdk.sh
|-- l1.sh
|-- l1app
|-- phycfg.xml
`-- phycfg_timer.xml
```

4.3 NB-IoT L2 Application

The NB-IoT L2 application is to be provided in binary form by the Layer 2 software vendor. For now, integration testing with Radisys* L2L3 is ongoing.

4.4 Test Applications

The FlexRAN NB-IoT package includes multiple supporting applications used for the execution of test scenarios. To use TestMAC:

1. Build the Wireless SDK component. The SDK package can be located anywhere in the file system. If it is located in `/home/turner/work/flexran/sdk`, issue the following command:

```
export DIR_WIRELESS_SDK_ROOT=/home/turner/work/flexran/sdk
```

For AVX2 systems (such as Broadwell):

```
export SDK_BUILD=build-avx2-icc
export WIRELESS_SDK_TARGET_ISA=avx2
export DIR_WIRELESS_SDK=${DIR_WIRELESS_SDK_ROOT}/${SDK_BUILD}
```

For AVX512 systems (e.g., Skylake):

```
export SDK_BUILD=build-avx512-icc
export WIRELESS_SDK_TARGET_ISA=avx512
export DIR_WIRELESS_SDK=${DIR_WIRELESS_SDK_ROOT}/${SDK_BUILD}
```

```
cd /home/turner/work/flexran/sdk/
./create-makefiles-linux.sh
```

For AVX2 systems:

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

For AVX512 systems:

```
cd build-avx512-icc
make install
```

2. Build TestMAC (simulation of NB-IoT L2).

```
cd /home/turner/work/flexran/build/l1driver_nbiot
./build.sh xclean
./build.sh
```

The resulting TestMAC elf file can be found under:

```
/home/turner/work/flexran/lte_l1driver/nbiot/
```

4.5 How to Start

This section covers how to preconfigure the host OS and start the guest (VM) OS, L1, and L2. It also provides a sample PHY application output.

4.5.1 Preconfigure Host OS

Reference host scripts can be found in Appendix A, Sections A.4.1, A.4.2, and A.4.3, respectively.

```
#login to the host OS
#cd /opt/vm/q2

#./config_host.sh
#./dpdk.sh
```

4.5.2 Start Guest (VM) OS

To start the guest (VM) OS:

```
./q_vm1_xd.sh
#The above command will start the VM, redirect the VM console on
the same shell where the script is launched.
Hit enter to get login prompt.
```

4.5.3 Start L1

To start L1, copy the L1 binaries and start-up scripts to the VM. In the example, the location of the L1 application is `/home/turner/work/`, and the location of the L2 application is `/home/turner/work/rsys/bundle/`.

The `dpdk.sh` script has to be updated with the correct PCIe bus information for a given PCIe NIC, as follows:

```
#
00:04.0 Ethernet controller: Intel Corporation 82599ES 10-Gigabit
SFI/SFP+ Network Connection (rev 01)
#
#cat ./dpdk.sh
…
$RTE_SDK/tools/dpdk_nic_bind.py --bind=igb_uio 0000:00:04.0
…

Modify phycfg.xml to use the same Ethernet port with correct
number of Cells
<Radio>
        <!-- Enable/disable radio                [0 - disable
(external app control radio), 1 - use Lib-Radio, 2 - use phy app
(obsolete)] -->
        <radioEnable>1</radioEnable>
        <!--  DPDK memory size allocated from hugepages [MB]
[default: 2048] -->
        <dpdkMemorySize>6144</dpdkMemorySize>
        <!--  DPDK Interrupt mode enable         [0 - disabled,
PMD is used, 1 - enabled, uio irq is used] -->
        <dpdkIrqMode>0</dpdkIrqMode>
        <!--  Ferry Bridge (FB) mode             [0 - LTE MODE, 1
- CPRI BYPASS MODE] -->
        <ferryBridgeMode>0</ferryBridgeMode>
        <!--  Number of Ethernet ports on FB     [0 - DPDK port 0,
1 - DPDK port 1, 2 - both DPDK port 0 and port 1 (CA mode with
two ETH)] -->
```

```
        <ferryBridgeEthPort>1</ferryBridgeEthPort>
        <!--  FB Synchronized CPRI ports        [0 - no reSync
REC & RE FPGA, 1 - reSync REC & REC FPGA] -->
        <ferryBridgeSyncPorts>0</ferryBridgeSyncPorts>
        <!--  FB Loopback Mode                  [0 - no optical
loopback connected REC<->RE, 1 - optical loopback connected REC<-
>RE] -->

<ferryBridgeOptCableLoopback>0</ferryBridgeOptCableLoopback>


        <!--  Radio Config 0 -->
        <RadioConfig0>
            <!-- DPDK: Add a PCI device in white list The
argument format is <[domain:]bus:devid.func> -->

<radioCfg0PCIeEthDev>0000:00:04.0</radioCfg0PCIeEthDev>
            <!-- DPDK: RX Thread core id [0-max core] -->
            <radioCfg0DpdkRx>1</radioCfg0DpdkRx>
            <!-- DPDK: TX Thread core id [0-max core] -->
            <radioCfg0DpdkTx>2</radioCfg0DpdkTx>
            <!--  Number of Tx Antenna         [1, 2, 4] -->
            <radioCfg0TxAnt>1</radioCfg0TxAnt>
            <!--  Number of Rx Antenna         [1, 2, 4] -->
            <radioCfg0RxAnt>1</radioCfg0RxAnt>
            <!--  Rx AGC configuration         [0 - Rx AGC
disabled, 1 - Rx AGC enabled (default for fpga release 1.3.1)] --
>
            <radioCfg0RxAgc>0</radioCfg0RxAgc>
            <!--  Number of cells running on this port   [1 -
Cell , 2 - Cells ] -->
            <radioCfg0NumCell>1</radioCfg0NumCell>
            <!-- First Phy instance ID mapped to this port    [0-
7 - valid range of PHY instance for first cell ] -->
            <radioCfg0Cell0PhyId>0</radioCfg0Cell0PhyId>
            <!-- Second Phy instance ID mapped to this port
[0-7 - valid range of PHY instance for second cell ] -->
            <radioCfg0Cell1PhyId>1</radioCfg0Cell1PhyId>
        </RadioConfig0>
```

Next, log on to the guest OS and run the following commands:

```
cd /home/turner/work/l1_sw/bin/lte/l1/lte_phy_nbiot
export export RTE_SDK=/opt/dpdk-17.11
export DIR_WIRELESS_SDK=/home/turner/work/sdk
./l1.sh
```

Note: After starting the L1 application, wait for at least 10 seconds before starting the L2 application.

Configuration of BBU pooling can be specified using `phycfg.xml`, as shown:

```
    <!-- CPU Binding to Application Threads -->
    <Threads>

         ...

        <!-- Wireless Subsystem Thread -->
        <wlsThread>6</wlsThread>

        <!-- DPDK Radio Master Thread -->
        <radioDpdkMaster>0</radioDpdkMaster>

        <!-- System Threads -->
        <systemThread>0</systemThread>

        <!-- Timer Thread -->
        <timerThread>0</timerThread>

        <!-- Enable L1 processing on Single Core per Cell -->
        <singleCore>0</singleCore>

        <!-- Frame Work Cores (Bit mask of all cores that are
used for BBU Pool in decimal) -->
        <bbuPoolCores>28</bbuPoolCores>
        <!-- Frame Work Core Priority -->
        <bbuPoolCorePriority>94</bbuPoolCorePriority>
        <!-- Frame Work Core Policy 0: SCHED_FIFO 1: SCHED_RR -->
        <bbuPoolCorePolicy>0</bbuPoolCorePolicy>
    </Threads>
```

4.5.4 Start L2

Starting L2 depends on the L2 vendor. Currently, integration with Radisys L2 testing is ongoing. This section serves as a placeholder. For additional information about starting L2, contact Radisys.

4.5.5 PHY Application Output Example

```
root@tm500-xeon-d-vm0:/lte_phy# ./l1.sh ./l1.sh
HugePages_Total:      14
HugePages_Free:       13
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:    1048576 kB
HugePages_Total:      14
HugePages_Free:       13
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:    1048576 kB
125260
kernel.sched_rt_runtime_us = -1
kernel.shmmax = 2147483648
kernel.shmall = 2147483648
error reading information on service irqbalance: No such file or
directory
./l1.sh: line 50: /proc/sys/kernel/nmi_watchdog: No such file or
directory
0000,00000000,00000000,00000000,00000001
0000,00000000,00000000,00000000,00000001
0000,00000000,00000000,00000000,00000001
0000,00000000,00000000,00000000,00000001
0000,00000000,00000000,00000000,00000001
0000,00000000,00000000,00000000,00000001
0000,00000000,00000000,00000000,00000001
0000,00000000,00000000,00000000,00000001
0000,00000000,00000000,00000000,00000001
0000,00000000,00000000,00000000,00000001
0000,00000000,00000000,00000000,00000001
0000,00000000,00000000,00000000,00000001
0000,00000000,00000000,00000000,00000001
0000,00000000,00000000,00000000,00000001
0000,00000000,00000000,00000000,00000001
0000,00000000,00000000,00000000,00000001
0000,00000000,00000000,00000000,00000001
0000,00000000,00000000,00000000,00000001
rmmod: ERROR: Module wls is not currently loaded
RADIO Mode
Unloading any existing DPDK UIO module
Loading DPDK UIO module
HOST
```

```
Network devices using DPDK-compatible driver
============================================
<none>

Network devices using kernel driver
===================================
0000:02:00.0 '82599ES 10-Gigabit SFI/SFP+ Network Connection'
if=eth32 drv=ixgbe unused=igb_uio
0000:02:00.1 '82599ES 10-Gigabit SFI/SFP+ Network Connection'
if=eth34 drv=ixgbe unused=igb_uio
0000:04:00.0 'I350 Gigabit Network Connection' if=eth31 drv=igb
unused=igb_uio
0000:04:00.3 'I350 Gigabit Network Connection' if=eth30 drv=igb
unused=igb_uio
0000:81:00.0 '82599ES 10-Gigabit SFI/SFP+ Network Connection'
if=eth33 drv=ixgbe unused=igb_uio
0000:81:00.1 '82599ES 10-Gigabit SFI/SFP+ Network Connection'
if=eth35 drv=ixgbe unused=igb_uio

Other network devices
=====================
<none>

Crypto devices using DPDK-compatible driver
===========================================
<none>

Crypto devices using kernel driver
==================================
<none>

Other crypto devices
====================
<none>
Unknown device: 0000:08:00.0. Please specify device in
"bus:slot.func" format
Unknown device: 0000:08:00.1. Please specify device in
"bus:slot.func" format

Network devices using DPDK-compatible driver
============================================
<none>

Network devices using kernel driver
```

```
==================================
0000:02:00.0 '82599ES 10-Gigabit SFI/SFP+ Network Connection'
if=eth32 drv=ixgbe unused=igb_uio
0000:02:00.1 '82599ES 10-Gigabit SFI/SFP+ Network Connection'
if=eth34 drv=ixgbe unused=igb_uio
0000:04:00.0 'I350 Gigabit Network Connection' if=eth31 drv=igb
unused=igb_uio
0000:04:00.3 'I350 Gigabit Network Connection' if=eth30 drv=igb
unused=igb_uio
0000:81:00.0 '82599ES 10-Gigabit SFI/SFP+ Network Connection'
if=eth33 drv=ixgbe unused=igb_uio
0000:81:00.1 '82599ES 10-Gigabit SFI/SFP+ Network Connection'
if=eth35 drv=ixgbe unused=igb_uio

Other network devices
=====================
<none>

Crypto devices using DPDK-compatible driver
===========================================
<none>

Crypto devices using kernel driver
==================================
<none>

Other crypto devices
====================
<none>
0
Command String =  0
using configuration file phycfg.xml
>> Running... ././l1app --cfgfile=phycfg.xml 0
========================
LTE PHY Application
========================
cline_get_string:Searching for string: cfgfile. Length of string:
7

cline_get_string:Found cfgfile: Val = phycfg.xml

Phycgf XML file parsed
Version found returned 0 6.2

--------------------------
```

```
PhyCfg.xml Version: 6.2
---------------------------

Apply config..
cline_print_info:Incomming settings:
 --version=6.2
 --mac2PhyBatchApi=1
 --phy2MacBatchApi=1
 --successiveNoApi=15
 --wls_dev_name=/dev/wls
 --dlIqLog=0
 --ulIqLog=0
 --iqLogDumpToFile=0
 --phyMlog=1
 --phyStats=1
 --radioEnable=1
 --dpdkMemorySize=1024
 --dpdkIrqMode=1
 --ferryBridgeMode=0
 --ferryBridgeEthPort=1
 --ferryBridgeSyncPorts=0
 --ferryBridgeOptCableLoopback=0
 --radioCfg0PCIeEthDev=0000:08:00.0
 --radioCfg0DpdkRx=1
 --radioCfg0DpdkTx=2
 --radioCfg0TxAnt=4
 --radioCfg0RxAnt=4
 --radioCfg0RxAgc=0
 --radioCfg1PCIeEthDev=0000:00:05.0
 --radioCfg1DpdkRx=5
 --radioCfg1DpdkTx=6
 --radioCfg1TxAnt=4
 --radioCfg1RxAnt=4
 --radioCfg1RxAgc=0
 --radioCfg2PCIeEthDev=0000:00:06.0
 --radioCfg2DpdkRx=5
 --radioCfg2DpdkTx=6
 --radioCfg2TxAnt=4
 --radioCfg2RxAnt=4
 --radioCfg2RxAgc=0
 --radioCfg3PCIeEthDev=0000:00:07.0
 --radioCfg3DpdkRx=5
 --radioCfg3DpdkTx=6
 --radioCfg3TxAnt=4
```

```
--radioCfg3RxAnt=4
--radioCfg3RxAgc=0
--radioCfg4PCIeEthDev=0000:00:08.0
--radioCfg4DpdkRx=5
--radioCfg4DpdkTx=6
--radioCfg4TxAnt=4
--radioCfg4RxAnt=4
--radioCfg4RxAgc=0
--radioPort0=0
--radioPort1=1
--radioPort2=0
--radioPort3=1
--taFiltEnable=1
--ircEnable=0
--mmseDisable=0
--pucchFormat2DetectThreshold=0
--prachDetectThreshold=100
--MlogSubframes=256
--MlogCores=8
--MlogSize=3084
--apiThread=2
--prachThread=4
--fftMainThread=3
--fftProc0Thread=3
--fftProc1Thread=4
--fftProc2Thread=5
--fftProc3Thread=6
--ifftMainThread=2
--ifftProc0Thread=2
--ifftProc1Thread=4
--ifftProc2Thread=5
--ifftProc3Thread=6
--dlMainThread=2
--dlProc0Thread=6
--dlProc1Thread=6
--dlProc2Thread=2
--dlProc3Thread=6
--ulMainThread=3
--ulProc0Thread=4
--ulProc1Thread=4
--ulProc2Thread=3
--ulProc3Thread=4
--wlsThread=2
```

```
 --radioDpdkMaster=0
 --systemThread=0
 --timerThread=0
 --singleCore=0

phycfg_apply: Initialize Radio Interface with Ferry Bridge
library
PhyCfg File completely read: 0
phycfg_apply:
PHY Logs Enabled:
phycfg_apply:-----------------
phycfg_apply:MLOG:                      YES
phycfg_apply:DL IQ:                     NO
phycfg_apply:UL IQ:                     NO
phycfg_apply:PHY STATS:                 YES
phycfg_apply:RADIO IQ DUMP TO FILE:     NO
phycfg_apply:FFT/IFFT on RADIO :        YES
phycfg_apply:-----------------
sys_load_app_in_ram:
sys_load_app_in_ram:AppInRam: Start: pid=125379
sys_load_app_in_ram:[err] AppInRam: Warning. Looked for 2 args in
'proc//maps' string, found: 2
sys_load_app_in_ram:[err] AppInRam: Warning. addr_end has to be
greater than addr_start
sys_load_app_in_ram:[err] AppInRam: Warning. Looked for 2 args in
'proc//maps' string, found: 2
sys_load_app_in_ram:[err] AppInRam: Warning. addr_end has to be
greater than addr_start
sys_load_app_in_ram:AppInRam: End. 0 areas checked to be loaded
in RAM

sys_init:Initialization
cmgr_init:initialization of console
System clock (rdtsc) resolution 2294693120 [Hz]
Ticks per us 2294 []
tlMlogInit: resource_freq: 2294, mlog_mask: 0xffffffff, filename:
mlog
set mlog_mask = 0xffffffff
set filename = mlog.bin
MLogOpen: filename(mlog.bin) mlogSubframes (256), mlogCores(8),
mlogSize(3084)
    mlogSubframes (256), mlogCores(8), mlogSize(3084)
 MLOG not opened!!!
MLog Storage: 0x7f6a05fc8010 -> 0x7f6a065cec3c
MLogInitializeMlogBuffer
```

```
Mlog Open successful
---------------------------------------
MLog Info: virt=0x00007f6a05fc8010 size=6319148
---------------------------------------
gpLteMac2PhyApiRecv:0x0x59cf180
wls_layer_init: WLS_Open /dev/wls
wls_lib: Open /dev/wls 0xc0085701
wls_lib: User Space Lib Context: us va 0x00007f6a05fb8000 kernel
va 0xffff880075db0000 pa 0x0000000075db0000 size 65536
wls_lib:
Mode 0
wls_lib:
WLS device /dev/wls [8]
wls_lib: hugePageSize on the system is 1073741824
wls_lib: shm open /tmp/phyappshm__dev_wls
wls_lib: Attach to shared memory
wls_lib: pvirtAddr  0x00002aaac0000000
wls_lib: WLS_Alloc: 0x00002aaac0000000 [1046478848]
wls_rx_handler:           [PID: 125380] binding on [CPU  2]
[PRIO: 96] [POLICY:  1]
di_radio_initdi_radio_cfg_setup nCC 1 radioItf 1
DPDK cores mask  0x7
DPDK memory size 1024
EAL: Detected 36 lcore(s)
EAL: Probing VFIO support...
num_ports 1
 ports[0] 0 ports[1] 0
DPDK cores 3
rte_eth_dev_count 0

Specified port number(1) exceeds total system port number(0)
valid_num_ports 1
 num_ports 0  ports[0] 0 ports[1] 0
RX_MBUF_POOL_PORT 0 size 640155648
TX_MBUF_POOL_PORT 0 size 320077824
Start of ltel1under timer mode
```

§

5 FlexRAN NB-IoT Setup Configuration

5.1 FlexRAN NB-IoT Supported Scenarios

The only verified and supported system configuration is one cell HD-FDD NB-IoT: UL and DL PHY test with TestMAC.

The validation is based on IQ data verification for downlink with a fixed TestMAC test configuration. For uplink, validation is based on uplink IQ data and fixed TestMAC test configuration. The tests cover L2-L1 API, NPSS, SSS, NPBCH, NPRACH, NPDSCH, PDSCCH, and NPUSCH physical processing.

Note: The PHY code is ported directly from NB-IoT small cell product, which already passed through the operator's field trial (end-to-end system, including UE and EPC) together with partner L2L3. For FlexRAN NB-IOT, integration tests with L2L3 are ongoing (Frequency Band 8).

Other scenarios are outside of the scope of this User Guide and are not supported. More information on features and limitations can be found in the *FlexRAN Reference Solution Software v1.5.0 Release Notes* (refer to Table 2).

5.2 Ferry Bridge

The current Intel Ferry Bridge FPGA module does not support the NB-IoT 1.92M sampling rate. Customers must implement their own NB-IoT FPGA module to support NB-IoT. The NB-IoT L1 application still supports the Ferry Bridge API (under bypass mode).

5.3 eNodeB

The eNodeB configuration is performed via the configuration file `wr_cfg.txt`, which is part of the Radisys* package for the L2 application.

Note: For now, Radisys L2 Integration testing with FlexRAN NB-IoT L1 is ongoing.

5.3.1 PHY Config

PHY is mainly configured by the configuration files. To support the USRP demo mode, a macro definition must be added when using PHY to connect to USRP.

5.3.1.1 Config File

The NB-IoT PHY configuration reuses the FlexRAN LTE PHY configuration file—`phycfg.xml`.

5.3.1.1.1 Timer Mode

```
...
<Radio>
        <radioEnable>0</radioEnable>
        <dpdkMemorySize>6144</dpdkMemorySize>
        <dpdkIrqMode>0</dpdkIrqMode>
        <ferryBridgeMode>0</ferryBridgeMode>
        <ferryBridgeEthPort>1</ferryBridgeEthPort>
        <ferryBridgeSyncPorts>0</ferryBridgeSyncPorts>

<ferryBridgeOptCableLoopback>0</ferryBridgeOptCableLoopback>

        <!--  Radio Config 0 -->
        <RadioConfig0>
         <radioCfg0PCIeEthDev>0000:00:04.0</radioCfg0PCIeEthDev>
            <radioCfg0DpdkRx>1</radioCfg0DpdkRx>
            <radioCfg0DpdkTx>2</radioCfg0DpdkTx>
            <radioCfg0TxAnt>2</radioCfg0TxAnt>
            <radioCfg0RxAnt>2</radioCfg0RxAnt>
            <radioCfg0RxAgc>0</radioCfg0RxAgc>
            <radioCfg0NumCell>1</radioCfg0NumCell>
            <radioCfg0Cell0PhyId>0</radioCfg0Cell0PhyId>
            <radioCfg0Cell1PhyId>1</radioCfg0Cell1PhyId>
        </RadioConfig0>
...
```

5.3.1.1.2 Ferry Bridge Mode (ByPass Mode)

Only Ferry Bridge bypass mode is supported. NB-IoT L1 SW implements iFFT/FFT/PRACH by itself, because current Ferry Bridge does not support 128 points iFFT/FFT and NB-IoT Prach processing.

```
<Radio>
        <radioEnable>1</radioEnable>
        <dpdkMemorySize>6144</dpdkMemorySize>
        <dpdkIrqMode>0</dpdkIrqMode>
        <ferryBridgeMode>1</ferryBridgeMode>
        <ferryBridgeEthPort>1</ferryBridgeEthPort>
        <ferryBridgeSyncPorts>0</ferryBridgeSyncPorts>
```

```
<ferryBridgeOptCableLoopback>0</ferryBridgeOptCableLoopback>

        <!--  Radio Config 0 -->
        <RadioConfig0>
         <radioCfg0PCIeEthDev>0000:00:04.0</radioCfg0PCIeEthDev>
            <radioCfg0DpdkRx>1</radioCfg0DpdkRx>
            <radioCfg0DpdkTx>2</radioCfg0DpdkTx>
            <radioCfg0TxAnt>1</radioCfg0TxAnt>
            <radioCfg0RxAnt>1</radioCfg0RxAnt>
            <radioCfg0RxAgc>0</radioCfg0RxAgc>
            <radioCfg0NumCell>1</radioCfg0NumCell>
            <radioCfg0Cell0PhyId>0</radioCfg0Cell0PhyId>
            <radioCfg0Cell1PhyId>1</radioCfg0Cell1PhyId>
        </RadioConfig0>
```

A detailed explanations of all `phycfg.xml` parameters can be found in *FlexRAN Reference Solution L1 XML Configuration User Guide* (refer to Table 2).

5.3.1.2 Macro Definition

The `makefile` is located at `/build/lte/l1app_nbiot`. Most of the OPT definitions are the same as with LTE. Only one `MACRO (NBIOT_USRP_MODE)` is used for NBIOT PHY. When this macro is enabled, PHY uses the USRP UHD API rather than the Ferry Bridge API.

5.4 EPC

FlexRAN NB-IoT is ported from Intel® Transcede™ brand NB-IoT L1, which passed through IODT with ZTE Commercial NB-IoT EPC. For the NB-IoT EPC simulator, contact Polaris Neworks* or Radisys*.

5.5 File Transfer Protocol (FTP)

General file transfer protocol (FTP) software can be used.

5.6 Commercial UE (CUE)

Intel suggests to use the following UE modules, because the NB-IoT small cell product passed IODT with the UEs. Also, FlexRAN NB-IOT PHY is ported directly from the small cell product. For FlexRAN NB-IOT IODT with UE, testing is still ongoing with Radisys L2L3.

- ZTEWeLink ME3612 NB-IoT chipset
- Huawei Boudica 120 NB-IoT chipset

5.7 RRH (SISO)

Contact the NB-IoT RRH vendors if you need to support NB-IoT bands. Band 8 is recommended.

For now, Intel uses NI USRP as an integration test setup with Radisys L2L3, because Ferry Bridge does not support NB-IoT. The Test Radio Band is Band 8. The mezzanine cards of USRP can support different band ranges. Intel suggests using the UBX-160 USRP mezzanine card.

To run NI USRP, copy `libuhd.so` to `/usr/local/lib`, making sure the path is configured in the `/etc/ld.so.conf` and `ldconfg` to make the path valid. For more details, refer to the *USRP Hardware Driver and USRP Manual* (refer to Table 2).

5.8 Cobham* TM500 Emulator

The TM500 LTE test equipment needs to be upgraded to support NB-IoT. For further information, consult with Cobham* support.

5.9 Recommended Start Sequence

The FlexRAN application performs the NB-IoT eNodeB functionality in terms of the NB-IoT network. The following sections describe the recommended start-up sequence for the setup.

5.9.1 Setup Network Components Except for eNodeB

Set up and enable all the network components that are external to the NB-IoT eNodeB according to the configurations described in the previous sections:

1. EPC
2. FTP server
3. TM500/CUEs
4. RRH
5. FPGA module for CPRI converter (similar in function to Ferry Bridge offload mode)

5.9.2 Start eNodeB

To start eNodeB:

1. Enter: `config host`
2. Enter: `config dpdk`
3. Start VM.

4. Enter ssh into the VM, and start L1.
5. Enter ssh into the VM and start L2. (The NB-IoT L2 start method depends on the L2 provider.)
6. Wait for TTI check marks to be printed on the L1 console.
7. While eNodeB is running, perform the attach procedure for all UEs.
8. Finally, start the traffic scenario according to your requirements.

5.9.3 Restarting System NB-IoT eNodeB

To restart the system NB-IoT eNodeB:

1. Exit from the L2 application. (Note: The NB-IoT L2 restart method depends on the L2 provider.)
2. Exit from the L1 application. (Type `exit` on the L1 console).
3. Power cycle Ferry Bridge.
4. Start eNodeB again, as described in Section 5.9.2.

Note: The VM does not need to be restarted between eNodeB sessions. Restarting the L2 and L1 applications is sufficient.

§

A Appendix

A.1 Recommended Kernel Configuration for Host and Guest OS

```
Wind River 6.0 build
recipes-kernel/linux/linux-windriver/config_baseline.cfg

CONFIG_LOCALVERSION="-WR6.0.0.25_preempt-rt"
#
# Power management and ACPI options
#
CONFIG_SUSPEND=y
CONFIG_SUSPEND_FREEZER=y
# CONFIG_HIBERNATION is not set
CONFIG_PM_SLEEP=y
CONFIG_PM_SLEEP_SMP=y
# CONFIG_PM_AUTOSLEEP is not set
# CONFIG_PM_WAKELOCKS is not set
# CONFIG_PM_RUNTIME is not set
CONFIG_PM=y
# CONFIG_PM_DEBUG is not set
# CONFIG_WQ_POWER_EFFICIENT_DEFAULT is not set
CONFIG_ACPI=y
CONFIG_ACPI_SLEEP=y
# CONFIG_ACPI_PROCFS is not set
# CONFIG_ACPI_PROCFS_POWER is not set
# CONFIG_ACPI_EC_DEBUGFS is not set
# CONFIG_ACPI_AC is not set
# CONFIG_ACPI_BATTERY is not set
CONFIG_ACPI_BUTTON=y
CONFIG_ACPI_VIDEO=m
CONFIG_ACPI_FAN=y
# CONFIG_ACPI_DOCK is not set
# CONFIG_ACPI_PROCESSOR is not set
# CONFIG_ACPI_IPMI is not set
CONFIG_ACPI_NUMA=y
CONFIG_ACPI_CUSTOM_DSDT_FILE=""
# CONFIG_ACPI_CUSTOM_DSDT is not set
```

```
# CONFIG_ACPI_INITRD_TABLE_OVERRIDE is not set
CONFIG_ACPI_BLACKLIST_YEAR=0
# CONFIG_ACPI_DEBUG is not set
# CONFIG_ACPI_PCI_SLOT is not set
# CONFIG_X86_PM_TIMER is not set
# CONFIG_ACPI_CONTAINER is not set
# CONFIG_ACPI_HOTPLUG_MEMORY is not set
# CONFIG_ACPI_SBS is not set
# CONFIG_ACPI_HED is not set
# CONFIG_ACPI_CUSTOM_METHOD is not set
# CONFIG_ACPI_APEI is not set
# CONFIG_SFI is not set
CONFIG_NO_HZ_FULL=y


#
# CPU Frequency scaling
#
# CONFIG_CPU_FREQ is not set

#
# CPU Idle
#
# CONFIG_CPU_IDLE is not set
# CONFIG_ARCH_NEEDS_CPU_IDLE_COUPLED is not set
CONFIG_PCI_IOV=y
CONFIG_CRYPTO_ZLIB=y
CONFIG_CRYPTO_LZO=y
```

CPU Frequency scaling and power control on CPU and PCIe were disabled to guarantee the response time for the real-time application.

A.2 Kernel Source on Guest OS

The kernel source code has to be installed in the guest OS to compile the `wls.ko` kernel module used by the FlexRAN application. The default build process for OVP6 can be modified to include the kernel sources in the VM image. Or, the Linux* kernel source code can be copied into the VM from the OVP6 build tree path: `Builds/ovp-guest/build/linux-windriver/linux`.

The destination for the kernel sources in the VM can be configured as per Linux kernel build scripts for any typical Linux system. For example, the following links can be set up to allow a successful build:

```
root@tm500-xeon-d-vm0:/lib/modules/3.10.87-ovp-rt93-
WR6.0.0.23_preempt-rt# ls -l
lrwxrwxrwx 1 root root    59 Dec  4  2015 build ->
/hdd2/kernels/kernel-3.10.87-ovp-rt93-wr6.0.0.23-preempt-rt
lrwxrwxrwx 1 root root    59 Aug 12 22:21 source ->
/hdd2/kernels/kernel-3.10.87-ovp-rt93-wr6.0.0.23-preempt-rt
```

A.3 Software Changes Required to Run the Artesyn* System

The three kernel changes outlined in this section are required on the host OS and guest OS in order to run WR OVP6 images on the Artesyn* system. The Wind River* Linux* build process can be updated to use the patches described in the following three sections to perform the correct configuration of the PCIe subsystem, according to the PCIe switch limitations of the Artesyn* Mac Core* chassis. Alternatively, the kernel can be updated later, after the installation, on a PCIe-7410 blade using the standard Linux approach for kernel changes.

A.3.1 pci_linux-suppress-cc-reset.patch

```
--- a/drivers/pci/quirks.c          2015-03-06 22:45:38.000000000
+0100
+++ b/drivers/pci/quirks.c          2015-11-26 15:48:12.856758394
+0100
@@ -3299,11 +3299,25 @@
    return 0;
 }

+static int reset_intel_82599ES(struct pci_dev *dev, int probe)
+{
+   if (probe)
+   {
+       dev_err (&dev->dev, "82599ES: Device reset suppressed\n");
+       return 1;
+   }


+   return 0;
+}
+

 #define PCI_DEVICE_ID_INTEL_82599_SFP_VF   0x10ed
 #define PCI_DEVICE_ID_INTEL_IVB_M_VGA      0x0156
 #define PCI_DEVICE_ID_INTEL_IVB_M2_VGA     0x0166
```

```
+#define PCI_DEVICE_ID_INTEL_82599ES_PF     0x10fb

 static const struct pci_dev_reset_methods
pci_dev_reset_methods[] = {
+   { PCI_VENDOR_ID_INTEL, PCI_DEVICE_ID_INTEL_82599ES_PF,
+         reset_intel_82599ES },
    { PCI_VENDOR_ID_INTEL, PCI_DEVICE_ID_INTEL_82599_SFP_VF,
          reset_intel_82599_sfp_virtfn },
    { PCI_VENDOR_ID_INTEL, PCI_DEVICE_ID_INTEL_IVB_M_VGA,
```

A.3.2 pci_linux-suppress-cc-save.patch

```
--- a/drivers/pci/pci.c          2015-11-26 16:45:50.480678305
+0100
+++ b/drivers/pci/pci.c          2015-11-26 16:48:40.855674358
+0100
@@ -1005,6 +1005,14 @@
 pci_save_state(struct pci_dev *dev)
 {
       int i;
+
+
+       if ((dev->vendor == PCI_VENDOR_ID_INTEL) &&
+           (dev->device == 0x10fb)) {
+               dev_err (&dev->dev, "ColetoCreek: save_state
suppressed\n");
+               return 0;
+       }
+
       /* XXX: 100% dword access ok here? */
       for (i = 0; i < 16; i++)
               pci_read_config_dword(dev, i * 4, &dev-
>saved_config_space[i]);
```

A.3.3 pci_qemu-pm-disable.patch

```
--- a/hw/i386/kvm/pci-assign.c      2015-01-01 06:02:11.000000000
+0000
+++ b/hw/i386/kvm/pci-assign.c      2015-01-01 06:02:49.000000000
+0000
@@ -360,6 +360,11 @@
     int pos = start ? start : PCI_CAPABILITY_LIST;
     int status;
```

```
+    if (cap == PCI_CAP_ID_PM) {
+   printf (" Device %x: PCI_CAP_ID_PM(%d) skipped\n", d->devfn,
cap);
+   return 0;
+    }
+
     status = assigned_dev_pci_read_byte(d, PCI_STATUS);
     if ((status & PCI_STATUS_CAP_LIST) == 0) {
         return 0;
```

A.4 Host Scripts

This section provides the scripts used in the host, including configuration, DPDK, and QEMU scripts.

A.4.1 config_host.sh

```
#cat  /opt/vm/config_host.sh
# *
# * Copyright 2009-2014 Intel Corporation All Rights Reserved.
# *
# * This program is free software; you can redistribute it and/or
modify
# * it under the terms of version 2 of the GNU General Public
License as
# * published by the Free Software Foundation.
# *
# * This program is distributed in the hope that it will be
useful, but
# * WITHOUT ANY WARRANTY; without even the implied warranty of
# * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU
# * General Public License for more details.
# *
# * You should have received a copy of the GNU General Public
License
# * along with this program; if not, write to the Free Software
# * Foundation, Inc., 51 Franklin St - Fifth Floor, Boston, MA
02110-1301 USA.
# * The full GNU General Public License is included in this
distribution
# * in the file called LICENSE.GPL.
# *
# * Contact Information:
```

```
# * Intel Corporation
# *

 /etc/init.d/glusterd stop
 /etc/init.d/openvswitch-controller stop
 /etc/init.d/openvswitch-switch stop
 /etc/init.d/libvirtd stop
 /etc/init.d/sanlock stop
 pkill wdmd
 /etc/init.d/nfsserver stop
 service auditd stop # now
 chkconfig auditd off # after reboot
 /etc/init.d/auditd stop
 pkill distccd
 /etc/init.d/crond stop

ulimit -c unlimited

MACHINE_TYPE=`uname -m`

if [ ${MACHINE_TYPE} == 'x86_64' ]; then

    ulimit -c unlimited
    echo 1 > /proc/sys/kernel/core_uses_pid
    sysctl -w kernel.sched_rt_runtime_us=-1
    sysctl -w kernel.sched_rt_period_us=-1
    for c in $(ls -d /sys/devices/system/cpu/cpu[0-9]*); do echo
performance >$c/cpufreq/scaling_governor; done
    echo 0 > /proc/sys/kernel/nmi_watchdog
    echo 1 >
/sys/module/rcupdate/parameters/rcu_cpu_stall_suppress

    for i in ` ls /proc/irq |grep -v default_smp_affinity | grep -
v 0 |grep -v 2 `
    do
           echo 1 > /proc/irq/$i/smp_affinity
    done



    for i in ` ls /proc/irq |grep -v default_smp_affinity | grep -
v 0 |grep -v 2 `
    do
           cat /proc/irq/$i/smp_affinity
    done
    sysctl -A | grep "sched" | grep -v "domain"
```

```
else
    echo "Machine type is not supported $MACHINE_TYPE"
    exit -1
fi

exit 0
```

A.4.2 dpdk.sh

```
#cat  /opt/vm/dpdk.sh

#! /bin/bash
export RTE_SDK=/opt/dpdk-17.11
export RTE_TARGET=x86_64-native-linuxapp-icc

#
# Unloads igb_uio.ko.
#
remove_igb_uio_module()
{
    echo "Unloading any existing DPDK UIO module"
    /sbin/lsmod | grep -s igb_uio > /dev/null
    if [ $? -eq 0 ] ; then
        sudo /sbin/rmmod igb_uio
    fi
}

#
# Loads new igb_uio.ko (and uio module if needed).
#
load_igb_uio_module()
{
    if [ ! -f $RTE_SDK/$RTE_TARGET/kmod/igb_uio.ko ];then
        echo "## ERROR: Target does not have the DPDK UIO Kernel
Module."
        echo "       To fix, please try to rebuild target."
        return
    fi

    remove_igb_uio_module

    /sbin/lsmod | grep -s uio > /dev/null
    if [ $? -ne 0 ] ; then
```

```
        if [ -f /lib/modules/$(uname -
r)/kernel/drivers/uio/uio.ko ] ; then
            echo "Loading uio module"
            sudo /sbin/modprobe uio
        fi
    fi

    # UIO may be compiled into kernel, so it may not be an error
if it can't
    # be loaded.

    echo "Loading DPDK UIO module"
    sudo /sbin/insmod $RTE_SDK/$RTE_TARGET/kmod/igb_uio.ko
    if [ $? -ne 0 ] ; then
        echo "## ERROR: Could not load kmod/igb_uio.ko."
        quit
    fi
}

load_igb_uio_module
```

A.4.3 q_vm1_xd.sh

A detailed description of QEMU Hypervisor configuration is available in the *Wind River* OVP6 User Guide* (refer to Table 2) and corresponding documentation.

```
#cat  /opt/vm/q_vm1_xd.sh

export VM_HOME=`pwd`
export AFFINITY_ARGS="taskset -c 1,2,3,4,5,6,7"
export QEMU=/usr/bin/qemu-system-x86_64
export DPDK_ARGS="-c 0x0f -n 4 --proc-type=secondary -- -enable-
dpdk"

/opt/dpdk-1.7.0/tools/dpdk_nic_bind.py --unbind 0000:09:00.0
/opt/dpdk-1.7.0/tools/dpdk_nic_bind.py --unbind 0000:09:00.1
/opt/dpdk-1.7.0/tools/dpdk_nic_bind.py --unbind 0000:08:11.2


$AFFINITY_ARGS $QEMU\
    -cpu host \
    -nographic \
    -k en-us \
    -m 20480 \
```

```
    -mem-path /mnt/huge \
    -mem-prealloc \
    -mlock \
    -name GUEST_1 \
    -enable-kvm \
    -smp cpus=7,cores=7,threads=1,sockets=1 \
    -vcpu 0,affinity=0x0002,prio=0 \
    -vcpu 1,affinity=0x0004,prio=95 \
    -vcpu 2,affinity=0x0008,prio=95 \
    -vcpu 3,affinity=0x0010,prio=95 \
    -vcpu 4,affinity=0x0020,prio=95 \
    -vcpu 5,affinity=0x0040,prio=95 \
    -vcpu 6,affinity=0x0080,prio=95 \
    -kernel $VM_HOME/vm1/bzImage_3.10.91-ovp-rt97-
WR6.0.0.25_preempt-rt \
    -append 'root=/dev/vda ro console=ttyS0 isolcpus=1-6
irqaffinity=0 nohz_full=0-6 clocksource=tsc tsc=perfect selinux=0
enforcing=0 default_hugepagesz=1G hugepagesz=1G hugepages=10
noswap idle=poll' \
    -drive file=$VM_HOME/vm1/wrlinux-image-ovp-kvm-x86-64-kvm-
guest.ext3,if=virtio \
    -drive file=$VM_HOME/vm1/disk_ph3.qcow2,if=none,id=drive-ide0-
0-0,format=qcow2 \
    -device ide-hd,bus=ide.0,unit=0,drive=drive-ide0-0-0,id=ide0-
0-0,bootindex=1 \
    -device pci-assign,host=0000:09:00.0,vcpuaffine \
    -device pci-assign,host=0000:09:00.1,vcpuaffine \
    -device pci-assign,host=0000:08:11.2,vcpuaffine \
    -fsdev local,security_model=passthrough,id=fsdev-fs0,path=/ \
    -device virtio-9p-pci,id=fs0,fsdev=fsdev-
fs0,mount_tag=hdd,bus=pci.0,addr=0x7
```

Note: The default OVP6 build does not include a virtualized HDD. It is possible to add this feature using a standard virtualization technique.

A.5 Build Server Preparation for WR-OVP6 Compilation

This section provides information regarding how to prepare to compile WR-OVP6.

A.5.1 Hardware Requirements for Build Server

The hardware requirements for build server are:

- At least 250 GB empty hard drive space for compilation
- At least 8 GB RAM
- Minimum processor spec 4 Core 2.4 GHz i5

Note: Allow 4 to 5 hours for the first build to complete.

A.5.2 Software Requirements

Ubuntu* 12.04.5 desktop version is known to work. Other distributions and versions of Linux* may also work. Refer to the Wind River OVP documentation for additional details and support (refer to Table 2).

A.5.3 Additional Packages

After successful installation of Ubuntu-12.04.5 LTS, run the following commands:

- `apt-get update`
- `apt-get install g++`
- `apt-get install diffstat`
- `apt-get install texinfo`
- `apt-get install gawk`
- `apt-get install chrpath`
- `apt-get install git`
- `apt-get install ccache`
- `apt-get install gcc-multilib`

Note: By default, the `/bin/sh` command is a symbolic link to the command `/bin/dash`. Redirect the link to `/bin/sh----> /bin/bash`.

- `# ln -s -f /bin/bash /bin/sh`

The following commands may be useful (depending on your development environment), but are not mandatory:

- `apt-get install openssh-server`
- `apt-get install screen`

§