如何在 NVIDIA Jetson Xavier NX 开发板上调试 AIROC™ CYW54591 - KBA235062

1 简介

本文档简要介绍了在 **NVIDIA Jetson Xavier NX 开发板**上调试 **AIROC™ CYW54591 Wi-Fi & Bluetooth®** 组合芯片的步骤。

1.1 操作要求

需了解 Linux 命令和基于 Linux 的文本编辑器，比如 nano/Vim

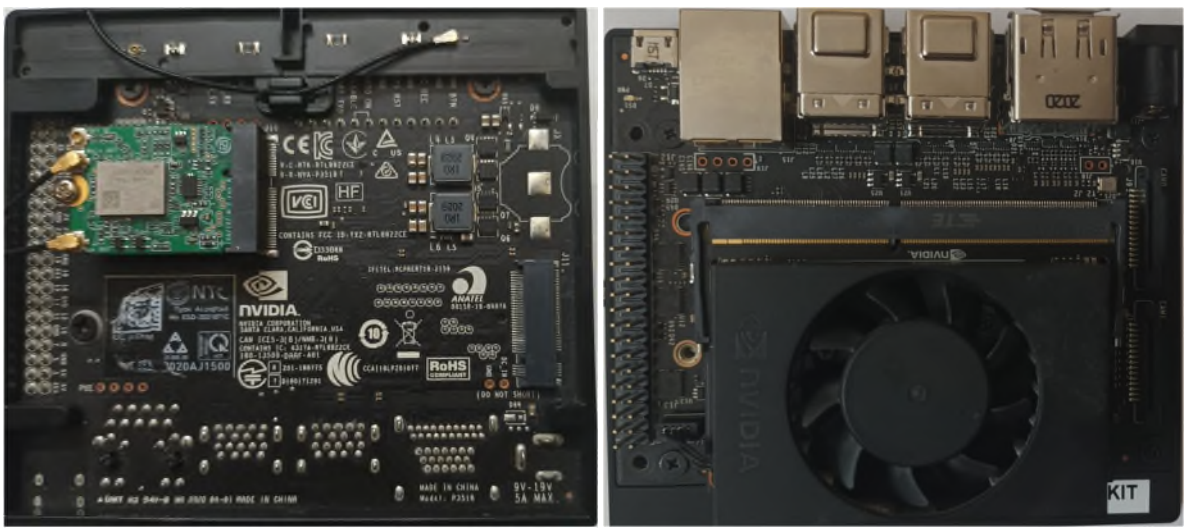
1.2 硬件要求

1. 英伟达 Jetson Xavier NX 开发板，必要的适配器和 USB 线
2. AIROC™ CYW54591 Wi-Fi 模块
3. 电脑 (Ubuntu 16.04.3 LTS)
4. HDMI 显示器
5. USB 键盘
6. USB 鼠标

Note: HDMI 显示器， USB 键盘和鼠标仅用于初期 bootup 创建用户名、密码。

2 硬件连接

- 如下图所示，将 AIROC™ CYW54591 Wi-Fi 芯片连接至 Jetson Xavier NX 开发板上 M.2 Key E Mini-PCIe slot (J10):



3 在 Jetson Xavier NX 开发板上调试 CYW54591

1. 从英伟达官网下载 Linux_for_Tegra (L4T) BSP 包，创建新文件夹，命名为“Jetpack”，将 Linux_for_Tegra (L4T) BSP 包解压并导入到 *Jetpack* 文件夹：

   ```
   $ mkdir Jetpack && cd Jetpack
   ```

2. 从下列地址下载并解压 *patches.zip* 文件：

   ```
   $ wget
   https://developer.nvidia.com/embedded/l4t/r32_release_v6.1/t186/jetson_linux_r32.6.1_aarch64.tbz2
   $ tar -xvf jetson_linux_r32.6.1_aarch64.tbz2
   ```

   ```
   user-muthu@usermuthu-ThinkPad-T460:~/Jetpack$ ls
   jetson_linux_r32.6.1_aarch64.tbz2  Linux_for_Tegra  patches
   user-muthu@usermuthu-ThinkPad-T460:~/Jetpack$
   ```

3. 下载示例根文件系统：

   ```
   $ wget
   https://developer.nvidia.com/embedded/l4t/r32_release_v6.1/t186/tegra_linux_sample-root-filesystem_r32.6.1_aarch64.tbz2
   ```

   ```
   user-muthu@usermuthu-ThinkPad-T460:~/Jetpack$ ls
   jetson_linux_r32.6.1_aarch64.tbz2  Linux_for_Tegra  patches  tegra_linux_sample-root-filesystem_r32.6.1_aarch64.tbz2
   user-muthu@usermuthu-ThinkPad-T460:~/Jetpack$
   ```

4. 安装 L4T 编译的必要编译环境：

   ```
   $ sudo apt-get --yes install git-core build-essential gcc-aarch64-linux-gnu
   ```

5. 下载内核和 bootloader 源码：

   ```
   export L4T_BASE_DIR=`pwd`/Linux_for_Tegra
   export DEFAULT_TAG_NAME="tegra-l4t-r32.6.1"
   export TAG_NAME="$DEFAULT_TAG_NAME"
   $  $L4T_BASE_DIR/source_sync.sh -t $TAG_NAME
   ```

6. 在内核 *defconfig* 文件中添加以下 Wi-Fi FMAC 补丁，禁用默认的 BCMDHD 驱动，启用其他 FMAC 相关的配置:

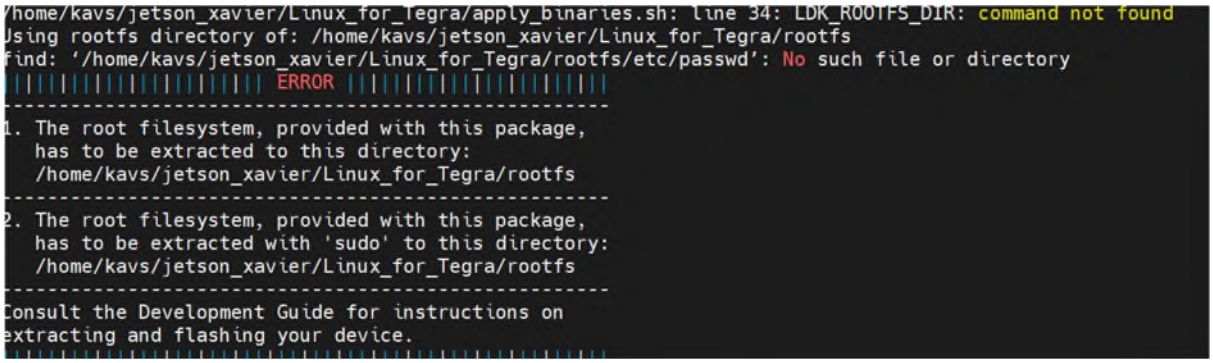
   ```
   export XAVIER_KERNEL_DIR=$L4T_BASE_DIR/sources/kernel/kernel-4.9
   export XAVIER_FMAC_DEFCONFIG_PATCH="./patches/xavier_nx_fmac_defconfig.patch"
   $ patch -d$XAVIER_KERNEL_DIR -p1 -N < $XAVIER_FMAC_DEFCONFIG_PATCH
   ```

7. 在 AARCH64 平台上交叉编译内核源码：

   ```
   export ARCH=arm64
   export CROSS_COMPILE=aarch64-linux-gnu-
   export LOCALVERSION=-tegra
   export KERNEL_SRC_PATH=$L4T_BASE_DIR/sources/kernel/kernel-4.9
   export TEGRA_KERNEL_OUT=$L4T_BASE_DIR/my_bins
   $  mkdir -p $TEGRA_KERNEL_OUT
   $  make -C $KERNEL_SRC_PATH O=$TEGRA_KERNEL_OUT tegra_defconfig && make -C
   $KERNEL_SRC_PATH O=$TEGRA_KERNEL_OUT -j$(nproc)
   ```

8. 将用户自定义包应用到 L4T 包的 rootfs：

```
export L4T_ROOTFS_DIR=$L4T_BASE_DIR/rootfs
$ sudo $L4T_BASE_DIR/apply_binaries.sh
```



如果系统报错，找到第二步中下载的 Jetpack 文件夹，解压示例 rootfs：

```
$ sudo tar -xvf tegra_linux_sample-root-filesystem_r32.6.1_aarch64.tbz2 -C
$L4T_ROOTFS_DIR
$ sudo $L4T_BASE_DIR/apply_binaries.sh
```

9. 生成并替换内核和 DTB：

```
$ cp $TEGRA_KERNEL_OUT/arch/arm64/boot/Image $L4T_BASE_DIR/kernel/.
$ cp $TEGRA_KERNEL_OUT/arch/arm64/boot/dts/* $L4T_BASE_DIR/kernel/dtb/.
```

10. 在 *rootfs* 路径下安装驱动模块：

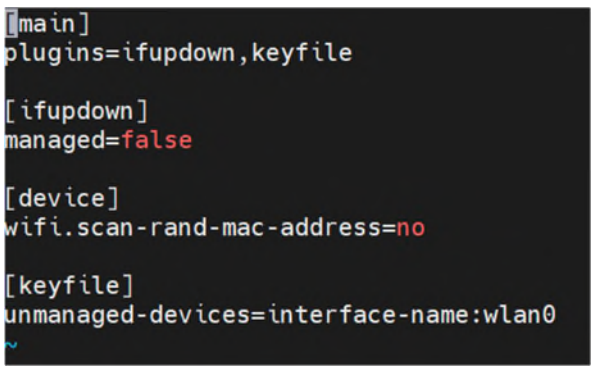
```
$ sudo make -C $KERNEL_SRC_PATH ARCH=arm64 O=$TEGRA_KERNEL_OUT LOCALVERSION=-tegra
INSTALL_MOD_PATH=$L4T_ROOTFS_DIR modules_install
```

11. 将内核配置文件(*.config*)拷贝到 *rootfs* 文件夹：

```
$ sudo cp $TEGRA_KERNEL_OUT/.config $L4T_ROOTFS_DIR/usr/src/linux-headers-4.9.253-
tegra-ubuntu18.04_aarch64/kernel-4.9/.config
```

12. 使用下列命令禁用控制 wlan0 界面的 NetworkManager 服务，打开以下文件 *$L4T_ROOTFS_DIR/etc/NetworkManager/NetworkManager.conf*，添加以下命令：

```
[keyfile]
unmanaged-devices=interface-name:wlan0
```



4 烧录 Jetson Xavier NX 开发板

先进入 Recovery 模式，再烧录 Jetson 开发板。

4.1 Jetson Xavier NX 进入 Recovery 模式

要进入 Recovery 模式，参考以下方式。

4.1.1 如果有控制台访问权限，登录 Jetson 平台

- 运行下列命令，进入 Recovery 模式：

```
$ sudo reboot --force forced-recovery
```

4.1.2 无控制台访问权限

1. 确保已关闭电源且未连接电源适配器。
2. 将 microSD 插入 Jetson Xavier NX [开发套件版本]模组 。
3. 用跳线帽短接 pin，进入强制 Recovery 模式，连接开发板排针[J14]上的第九针[GND]和第十针[FC REC]。
4. 使用 USB Micro-B 接口连接主机与与开发板。
5. 将电源适配器接入电源插孔[J16]。

   强制 Recovery 模式下，设备会自动开机。
6. 在主机终端运行 `lsusb` 命令。

   如果 Jetson 开发板已进入 Recovery 模式，终端上会显示以下输出：

   ```
   Bus 001 Device 010: ID 0955:7e19 NVidia Corp.
   ```
7. 断开跳线帽，退出强制 Recovery 模式。

4.2 用系统镜像烧录开发板

- 运行以下命令

```
$ cd $L4T_BASE_DIR
$ sudo ./flash.sh jetson-xavier-nx-devkit mmcblk0p1
```

刷写镜像需要花费若干分钟，完成后，Jetson Xavier NX 套件会自动重启。

4.3 创建用户名和密码

创建用户名和密码需要使用 HDMI 显示器、USB 键盘和鼠标。

1. 点击以下链接：
   https://developer.nvidia.com/embedded/learn/get-started-jetson-xavier-nx-devkit#setup
2. 要验证 Jetson Xavier NX 的主机 PCIe 控制器是否能检测到 AIROC™ CYW54591 芯片，在目标设备上运行 `lspci` 命令：

```
ifx@ifxhost:~$ lspci
0004:00:00.0 PCI bridge: NVIDIA Corporation Device 1ad1 (rev a1)
0004:01:00.0 Network controller: Broadcom Inc. and subsidiaries Device 4417 (rev 0d)
```

4.4 在 Jetson Xavier 开发板上编译 FMAC 驱动

1. 点击下列链接，下载 *cypress-fmac-v5.10.9-2021_1020.zip* 文件，在 Xavier NX 开发板中创建新文件夹，将下载的文件拷贝到该文件夹。使用 WinSCP 或基于 FTP 的软件。
   https://community.infineon.com/t5/Wi-Fi-Bluetooth-for-Linux/Cypress-Linux-WiFi-Driver-Release-FMAC-2021-10-20/m-p/322639#M2155
2. 解压缩上一步中下载的文件：

```
$ unzip cypress-fmac-v5.10.9-2021_1020.zip
```

可得下列文件:

```
cypress-backports-v5.10.9-2021_1020-module-src.tar.gz
cypress-cirrent-1.60.tar.gz
cypress-devicetree-2021-10-20.tar.gz
cypress-firmware-v5.10.9-2021_1020.tar.gz
cypress-hostap_2_9-1-2021_1020.tar.gz
cypress-patch-v5.10.9-2021_1020.tar.gz
```

3. 在解压缩的文件中提取 *cypress-backports-v5.10.9-2021_1020-module-src.tar.gz* 文件:

   ```
   $ tar -xvf cypress-backports-v5.10.9-2021_1020-module-src.tar.gz
   ```

4. 建立路径，将内核头文件输出至第三步中提取的文件*(v5.10.9-backports)*:

   ```
   $ export MY_KERNEL=/usr/src/linux-headers-4.9.253-tegra-
   ubuntu18.04_aarch64/kernel-4.9/
   ```

5. 从 *defconfigs/brcmfmac* 中指定的 FMAC 驱动配置生成配置文件（*.config*）：

   ```
   $ make KLIB=$MY_KERNEL KLIB_BUILD=$MY_KERNEL defconfig-brcmfmac
   ```

6. 编译，生成内核模块:

   ```
   $ make KLIB=$MY_KERNEL KLIB_BUILD=$MY_KERNEL modules
   ```

   内核模块可见于下列路径:

   ```
   ./compat/compat.ko
   ./net/wireless/cfg80211.ko
   ./drivers/net/wireless/broadcom/brcm80211/brcmutil/brcmutil.ko
   ./drivers/net/wireless/broadcom/brcm80211/brcmfmac/brcmfmac.ko
   ```

7. 将内核模块组复制到新文件夹后，重启设备。

   该文件夹通常命名为 “kernel_module” 。

8. 解压缩 *cypress-firmware-v5.10.9-2021_1020.tar.gz*:

   ```
   $ tar -xvf cypress-firmware-v5.10.9-2021_1020.tar.gz
   ```

   ```
   cyfmac43012-sdio.bin        cyfmac43439-sdio.bin        cyfmac4356-pcie.clm_blob    cyfmac4373-sdio.clm_blob        LICENCE
   cyfmac43012-sdio.clm_blob   cyfmac43439-sdio.clm_blob   cyfmac4356-sdio.bin         cyfmac4373-sdio.industrial.bin  README
   cyfmac43340-sdio.bin        cyfmac43455-sdio.bin        cyfmac4356-sdio.clm_blob    cyfmac4373-usb.bin              versions
   cyfmac43362-sdio.bin        cyfmac43455-sdio.clm_blob   cyfmac43570-pcie.bin        cyfmac54591-pcie.bin
   cyfmac4339-sdio.bin         cyfmac4354-sdio.bin         cyfmac43570-pcie.clm_blob   cyfmac54591-pcie.clm_blob
   cyfmac43430-sdio.bin        cyfmac4354-sdio.clm_blob    cyfmac4373.clm_blob         cyfmac54591-sdio.bin
   cyfmac43430-sdio.clm_blob   cyfmac4356-pcie.bin         cyfmac4373-sdio.bin         cyfmac54591-sdio.clm_blob
   ```

Note: *若*/lib/firmware/cypress *目录已存在，忽略此步骤。*

9. 在*/lib* 路径下创建新文件夹，命名为“firmware”，再在*/firmware* 路径下创建新文件夹，命名为“cypress”：

   ```
   $ sudo mkdir -p /lib/firmware/cypress
   ```

10. 将以下命令拷贝至 */lib/firmware/cypress*:

    - *cyfmac54591-pcie.bin*
    - *cyfmac54591-pcie.clm_blob*
    - *cyfmac54591-pcie.txt*

Note: *cyfmac54591-pcie.txt 是 NVRAM 文件，不包含于 FMAC 发布数据包中，可从模块供应商处获取。1XA(54591)模块的 NVRAM 文件可在以下 Murata GitHub 链接中下载:* ***https://github.com/murata-wireless/cyw-fmac-nvram/blob/master/cyfmac54591-pcie.1XA.txt***

11. 点开第七步创建的 *kernel_module* 文件夹，加载内核模块。加载所有模块前，如果已加载 *cfg80211.ko* 和 *compat.ko*，卸载该模块：

```
sudo rmmod cfg80211.ko
sudo rmmod compat.ko
$  sudo insmod compat.ko
$ sudo insmod cfg80211.ko
$ sudo insmod brcmutil.ko
$ sudo insmod brcmfmac.ko
```

加载模块时应严格按照上述顺序，卸载模块时按照相反的顺序。

12. 验证模块是否完成加载，使用以下代码：

```
$ lsmod
```

```
Module                  Size  Used by
brcmfmac              362179  0
brcmutil               12155  1 brcmfmac
bnep                   19078  2
fuse                  117972  3
zram                   29689  4
xt_conntrack            3979  1
ipt_MASQUERADE          2634  1
nf_nat_masquerade_ipv4     3993  1 ipt_MASQUERADE
nf_conntrack_netlink    33288  0
nfnetlink               9780  2 nf_conntrack_netlink
xt_addrtype             3915  2
iptable_filter          3008  1
iptable_nat             3423  1
nf_conntrack_ipv4      14222  2
nf_defrag_ipv4          2129  1 nf_conntrack_ipv4
nf_nat_ipv4             8176  1 iptable_nat
nf_nat                 25404  2 nf_nat_masquerade_ipv4,nf_nat_ipv4
nf_conntrack          132473  6 nf_conntrack_ipv4,nf_conntrack_netlink,nf_nat_masquerade_ipv4,xt_conntrack,nf_nat_ipv4,nf_nat
br_netfilter           17588  0
overlay                53801  0
nvgpu                1736376  23
userspace_alert         6833  0
cfg80211              835793  1 brcmfmac
compat                122146  2 brcmfmac,cfg80211
ip_tables              21421  2 iptable_filter,iptable_nat
x_tables               38080  5 ip_tables,iptable_filter,ipt_MASQUERADE,xt_addrtype,xt_conntrack
```

13. 使用静态 IP 地址配置网络接口:

```
$ sudo ifconfig wlan0 192.168.1.2 up
```

给 AP 的子网分配一个对应的静态 IP 地址。

14. 使用以下命令验证网络接口:

```
$ ifconfig
```

```
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.2  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::7274:14ff:fe12:3826  prefixlen 64  scopeid 0x20<link>
        ether 70:74:14:12:38:26  txqueuelen 1000  (Ethernet)
        RX packets 2315  bytes 199954 (199.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2922  bytes 248392 (248.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

4.5 将 WLAN 配置为 SoftAP

在用户空间 Linux 守护进程中，推荐使用 *hostapd* 将 WLAN 配置为 SoftAP。*hostapd* 实现了 IEEE 802.11 协议的 AP 管理，支持 IEEE 802.1X/WPA/WPA2/WPA3/EAP 授权、RADIUS 客户端、EAP 服务器和 RADIUS 授权服务器。通过不同配置，*hostapd* 可在以上任一模式中执行。最初，*hostapd* 只用作守护进程，现也支持前端程序，如子程序 *hostapd_cli*。

1. 使用以下命令，安装 *hostapd*:

   ```
   $ sudo apt-get install hostapd
   ```

   安装完成后，自动创建 */etc/hostapd* 文件夹。

   ```
   $ cd /etc/hostapd
   $ sudo vi hostapd.conf
   ```

   例:

   ```
   # Hostapd configuration file for open security 5G 80MHz.
   interface=wlan0
   driver=nl80211
   ctrl_interface=/var/run/hostapd
   ssid=JETSON_XAVIER
   hw_mode=a
   channel=36
   ignore_broadcast_ssid=0
   wmm_enabled=1
   ieee80211ac=1
   vht_oper_chwidth=1
   vht_oper_centr_freq_seg0_idx=42
   ht_capab=[SHORT-GI-20][SHORT-GI-40][HT40+][HT40-]
   vht_capab=[SHORT-GI-80]
   ```

2. 使用以下命令，创建 AP，命名为“JETSON_XAVIER”：

   ```
   $ sudo hostapd ./hostapd.conf -B -dd
   ```

   `-dd` 用于启用调试打印功能。

3. 查看 AP 状态:

   ```
   $ sudo hostapd_cli status
   ```

   如果出现类似以下的错误，运行下列命令，关闭 *hostapd*:

   ```
   ifx@ifx-desktop:/etc/hostapd$ sudo hostapd ./hostapd.conf -B
   Configuration file: ./hostapd.conf
   ctrl_iface exists and seems to be in use - cannot override it
   Delete '/var/run/hostapd/wlan0' manually if it is not used anymore
   Failed to setup control interface for wlan0
   wlan0: Unable to setup interface.
   wlan0: interface state UNINITIALIZED->DISABLED
   wlan0: AP-DISABLED
   hostapd_free_hapd_data: Interface wlan0 wasn't started
   nl80211: deinit ifname=wlan0 disabled_11b_rates=0
   ifx@ifx-desktop:/etc/hostapd$
   ```

   ```
   $ sudo rm -r /var/run/hostapd
   ```

4.6 将 WLAN 配置为 STA

推荐使用 *wpa_supplicant* 。*wpa_supplicant* 是用户空间 Linux 守护进程，用于处理 STA 的 WLAN 认证、关联、断开关联、取消认证过程。与 hostapd 一样，配置 *wpa_supplicant* 的过程很重要。

配置 wpa_supplicant 需要使用文本文件，该文件列出了所有接受的网络和安全策略，包括预共享密钥。该配置文件中的所有文件路径应使用完整的（绝对路径，而不是相对于工作目录的）路径，以允许修改工作目录。

下列为使用 *wpa_supplicant* 配置开放网络的代码示例：

```
# WPA Supplicant with open network security configuration.
ctrl_interface=/var/run/wpa_supplicant
update_config=1
device_name=JETSON-LINUX
device_type=10-0050F204-5
config_methods=virtual_push_button physical_display keyboard
interworking=1

network={
        ssid="[redacted]"
        key_mgmt=NONE
}
```

1. 各种 *wpa_supplicant* 配置，参阅 */etc/wpa_supplicant.conf*:

   ```
   $ sudo vi /etc/wpa_supplicant1.conf
   ```

2. 复制上述代码示例，配置 *wpa_supplicant* 构建开放网络。

3. 完成配置后，启动 *wpa_supplicant*，连接到你的办公室或家庭接入点，继续配置，运行以下命令:

   ```
   $ sudo wpa_supplicant -B -i wlan0 -c /etc/wpa_supplicant1.conf
   ```

   在这行命令中，

   `-B` 用来在后台运行 wpa_supplicant 守护进程

   `-c` 选项用于提供配置文件，在本例中，配置文件是，*wpa_supplicant1.conf*

   `-i` 选项用于选择网络接口

4. 运行以下命令，检查连接的状态:

   $ sudo wpa_cli status

```
Selected interface 'wlan0'
bssid=cc:46:4e:cd:d4:9e
freq=2437
ssid=[redacted]
id=0
mode=station
pairwise_cipher=NONE
group_cipher=NONE
key_mgmt=NONE
wpa_state=COMPLETED
ip_address=192.168.43.4
address=70:74:14:12:38:26
uuid=728b4d1b-4f94-5f78-b5ce-ad683e0dd7e3
```

5. 如果出现类似以下的错误，运行以下命令，关闭 *wpa_supplicant*:

```
$ sudo rm -r /var/run/wpa_supplicant/
removing /var/run/wpa_supplicant/
```

```
ifx@ifx-desktop:/etc$ sudo wpa_supplicant -B -i wlan0 -c /etc/wpa_supplicant1.conf
Successfully initialized wpa_supplicant
ctrl_iface exists and seems to be in use - cannot override it
Delete '/var/run/wpa_supplicant/wlan0' manually if it is not used anymore
Failed to initialize control interface '/var/run/wpa_supplicant'.
You may have another wpa_supplicant process already running or the file was
left by an unclean termination of wpa_supplicant in which case you will need
to manually remove this file before starting wpa_supplicant again.

nl80211: deinit ifname=wlan0 disabled_11b_rates=0
```

4.7 Ping STA 和 SoftAP

Ping STA 和 SoftAP，检查连接状况。

1. STA, 运行以下命令:

```
$ ping -I wlan0 <IP address of softAP>
```

2. SoftAP, 运行以下命令:

```
$ ping -I wlan0 <IP address of STA>
```

5 吞吐量测试

5.1 TCP 吞吐量测试

1. 在 Device-1-TCP 服务器（SoftAP 接口）上运行以下命令:

```
$ iperf -s -f m -i 1 -w 8m -l 8k
```

2. 在 Device-2-TCP 服务器（STA 接口）上运行以下命令:

```
$ iperf -c <TCP Server's WLAN IP address> -i 1 -f m -t 30 -w 8m -l 8k
```

5.2 UDP 吞吐量测试

1. 在 Device-1-UDP 服务器（SoftAP 接口）上运行以下命令:

```
$ iperf -s -u -i 1
```

2. 在 Device-2-UDP 服务器（STA 接口）上运行以下命令:

```
$ iperf -c<UDP Server's WLAN IP address> -u -i 1 -f m -t 30 -b 1200m
```

Revision history

Document version	Date of release	Description of changes
**	2023-03-07	译自英文 KBA235062 rev. *

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition 2022-03-22

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about this document?

Go to https://community.infineon.com

Document reference

002-35062 Rev. **

IMPORTANT NOTICE

The information contained in this document is given as a hint for the implementation of the product only and shall in no event be regarded as a description or warranty of a certain functionality, condition or quality of the product. Before implementation of the product, the recipient of this application note must verify any function and other technical information given herein in the real application. Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this application note.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on the product, technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies office (**www.infineon.com**).

WARNINGS

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.