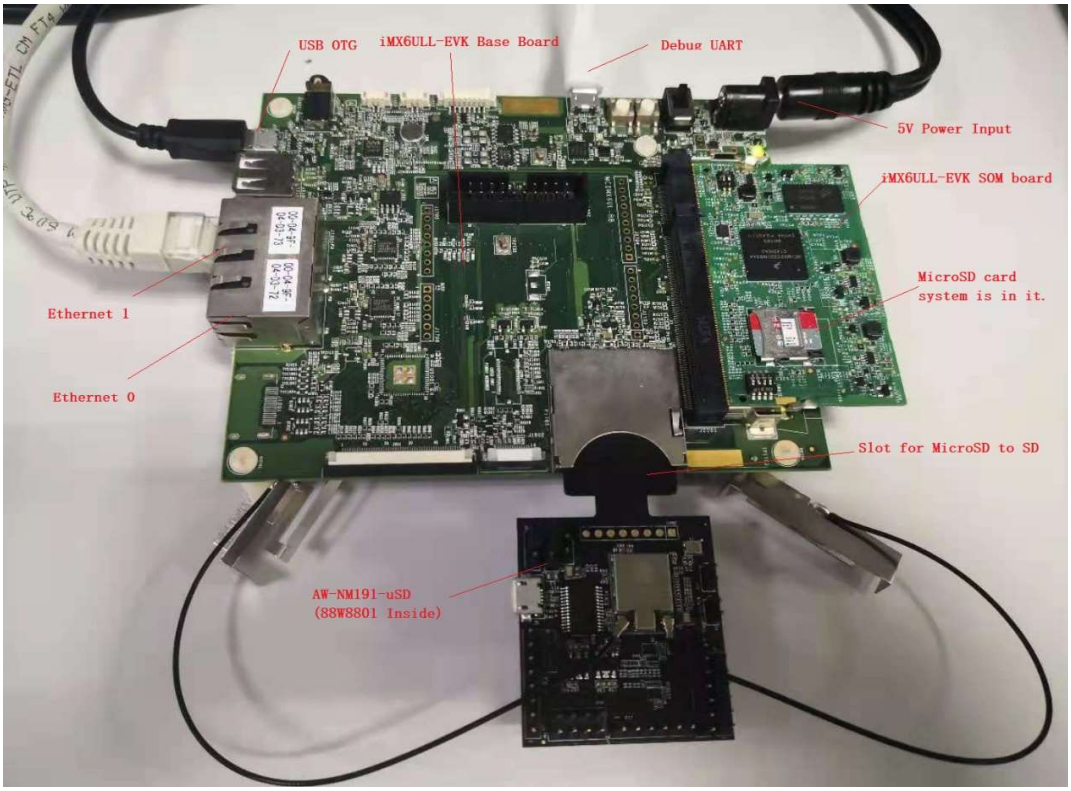
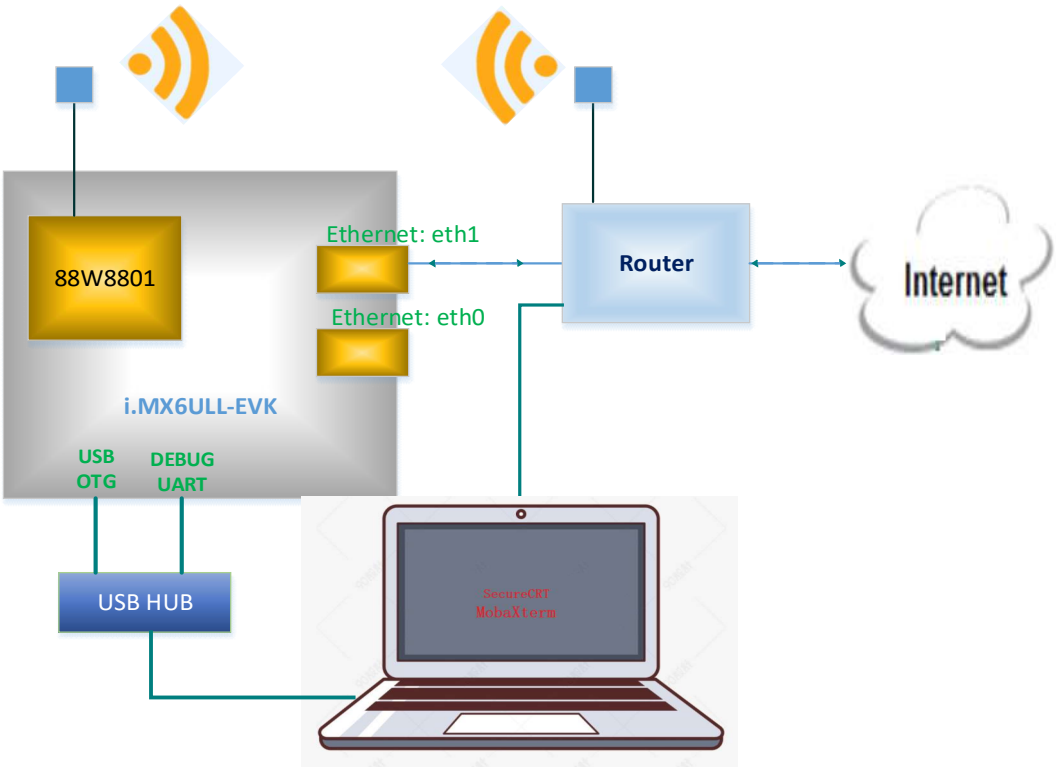
How to integrate Linux driver of 88W8801 To i.MX6ULL-EVK And L5.4.70_2.3.0 BSP





This article will describe how to integrate 88W8801 linux driver to i.MX6ULL-EVK platform in the following steps.

Before beginning our job, ubuntu 20.04 LTS should be installed on computer or VMWare player. User's can refer to **Linux_BSP_compilation_for_iMX_platform.pdf** to prepare linux eviroment.

Step 1: Exporting the cross-compilation tool chain from Yocto BSP
Step 2: Downloading Demo Image of DEMO i.MX6ULL-EVK from NXP website
Step 3: Downloading uuu tool
Step 4: Downloading Demo Images into MircroSD card of i.MX6ULL-EVK by uuu tool
Step 5: Booting i.MX6ULL-EVK board
Step 6: Downloading 88W8801 Linux driver and cross compiling it.
Step 7: Loading Linux driver of 88W8801 on i.MX6ULL-EVK
Step 8: Configuring 88W8801 and accessing Internet

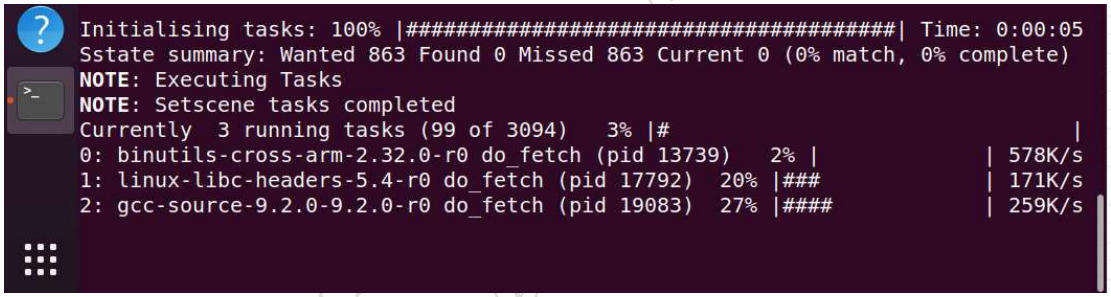
Step 1: Exporting the cross-compilation tool chain from Yocto BSP

- **Getting Yocto Source**

```
# mkdir ~/bin (this step may not be needed if the bin folder already exists)
# curl https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
# chmod a+x ~/bin/repo
# export PATH=~/bin:$PATH
# mkdir imx-yocto-bsp-l5.4.70_2.3.0
# cd imx-yocto-bsp-l5.4.70_2.3.0
# repo init -u https://source.codeaurora.org/external/imx/imx-manifest -b imx-linux-zeus -m imx-5.4.70-2.3.0.xml
# repo sync
```

- **Exporting cross compile tool chain for arm v7**

```
# DISTRO=fsl-imx-fb MACHINE=imx6qpsabreauto source imx-setup-release.sh -b build-fb
# cd ../
# DISTRO=fsl-imx-fb MACHINE=imx6qpsabreauto bitbake core-image-minimal -c populate_sdk
```



After all tasks are done, a .sh file will be find in build_fb/tmp/deploy/sdk, it's name is like this:

fsl-imx-fb-glibc-x86_64-core-image-minimal-cortexa9t2hf-neon-toolchain-5.4-zeus.sh

copy the file to /opt,

cd build_fb/tmp/deploy/sdk

sudo cp **fsl-imx-fb-glibc-x86_64-core-image-minimal-cortexa9t2hf-neon-toolchain-5.4-zeus.sh /opt**

cd /opt

chmod a+x **fsl-imx-fb-glibc-x86_64-core-image-minimal-cortexa9t2hf-neon-toolchain-5.4-zeus.sh**

sudo ./ **fsl-imx-fb-glibc-x86_64-core-image-minimal-cortexa9t2hf-neon-toolchain-5.4-zeus.sh**

Use the default directory to install it, and the tool chain installation will be done.

Step 2: Downloading Demo Image of DEMO i.MX6ULL-EVK from NXP website

Link is :

https://www.nxp.com/design/software/embedded-software/i-mx-software/embedded-linux-for-i-mx-applications-processors:IMXLINUX?tab=Design_Tools_Tab

Create a directory for demo image, please! For example, **e:/imx6ull-demo-image**, the copy demo image to the directory, and decompress it.

imx_mcore_demos	2020/12/21 9:33	文件夹	
samples	2020/12/21 9:33	文件夹	
EULA	2020/12/21 9:32	文本文档	37 KB
fsl-image-mfgtool-initramfs-imx_mfg...	2020/12/21 9:33	U-BOOT 文件	14,569 KB
GPLv2	2020/12/21 9:32	文件	19 KB
imx6ul-9x9-evk.dtb	2020/12/21 9:32	DTB 文件	35 KB
imx6ul-9x9-evk-btwifi.dtb	2020/12/21 9:32	DTB 文件	36 KB
imx6ul-9x9-evk-btwifi-oob.dtb	2020/12/21 9:32	DTB 文件	36 KB
imx6ul-9x9-evk-ldo.dtb	2020/12/21 9:32	DTB 文件	35 KB
imx6ul-14x14-evk.dtb	2020/12/21 9:32	DTB 文件	35 KB
imx6ul-14x14-evk-btwifi.dtb	2020/12/21 9:32	DTB 文件	36 KB
imx6ul-14x14-evk-btwifi-oob.dtb	2020/12/21 9:32	DTB 文件	36 KB
imx6ul-14x14-evk-csi.dtb	2020/12/21 9:32	DTB 文件	35 KB
imx6ul-14x14-evk-ecspi.dtb	2020/12/21 9:32	DTB 文件	35 KB
imx6ul-14x14-evk-ecspi-slave.dtb	2020/12/21 9:32	DTB 文件	35 KB
imx6ul-14x14-evk-emmc.dtb	2020/12/21 9:32	DTB 文件	35 KB
imx6ul-14x14-evk-gpmi-weim.dtb	2020/12/21 9:32	DTB 文件	36 KB
imx6ull-9x9-evk.dtb	2020/12/21 9:32	DTB 文件	35 KB
imx6ull-9x9-evk-btwifi.dtb	2020/12/21 9:32	DTB 文件	36 KB
imx6ull-9x9-evk-btwifi-oob.dtb	2020/12/21 9:32	DTB 文件	36 KB
imx6ull-9x9-evk-ldo.dtb	2020/12/21 9:32	DTB 文件	35 KB
imx6ull-14x14-evk.dtb	2020/12/21 9:32	DTB 文件	35 KB
imx6ull-14x14-evk-btwifi.dtb	2020/12/21 9:32	DTB 文件	36 KB
imx6ull-14x14-evk-btwifi-oob.dtb	2020/12/21 9:32	DTB 文件	36 KB
imx6ull-14x14-evk-emmc.dtb	2020/12/21 9:32	DTB 文件	35 KB
imx6ull-14x14-evk-gpmi-weim.dtb	2020/12/21 9:32	DTB 文件	36 KB
imx6ulz-14x14-evk.dtb	2020/12/21 9:32	DTB 文件	29 KB
imx6ulz-14x14-evk-btwifi.dtb	2020/12/21 9:32	DTB 文件	30 KB
imx6ulz-14x14-evk-emmc.dtb	2020/12/21 9:32	DTB 文件	29 KB

Step 3: Downloading uuu tool

The link is :

https://github.com/NXPmicro/mfgtools/releases

uuu	3.96 MB
uuu.exe	1.24 MB
UUU.pdf	678 KB
uuu_mac	527 KB
uuu_source-1.4.127.tar.gz	1.49 MB
uuu_source-1.4.127.zip	1.84 MB
Source code (zip)	
Source code (tar.gz)	

uuu.exe is for windows 10, uuu is for linux(ubuntu). After downloading it, copy uuu.exe to **e:/imx6ull-demo-image,** uuu.exe will in the same directory as images of i.mx6ull-evk.

Step 4: Downloading Demo Images into MircroSD card of i.MX6ULL-EVK by uuu tool

- **Setting boot mode on SOM board**

Table 7. Booting from TF on i.MX 6UltraLite EVK and i.MX 6ULL EVK

Switch	D1	D2	D3	D4
SW601	OFF	OFF	ON	OFF
SW602	ON	OFF	-	-

For SW602,

D1—ON; D2—OFF, Internal boot

D1—OFF; D2—ON , Serial Download mode (download image from USB OTG)

If microSD card is empty(no images), even if BOOT MODE is set to be internal boot, internal ROM CODE of CPU will also jump to Serial Download Mode.

- **Preparing a MicroSD card with 8GB size and insert it to slot on SOM board.**
- **Connecting USB OTG and debug UART to USB HUB**

  Connect USB OTG and debug UART to USB HUB with USB OTG cables, then Power On board with 5V adapter.

  --For USB OTG, HID compliance device will be showed in device manager.

  --For Debug UART, Silicon Labs CP210x USB to UART Bridge(COM15) in device manager. (On my computer, it is COM15)

- Starting window command line, and enter **e:/imx6ull-demo-image**
- Running the command to begin to download images to microSD card on SOM board.

  # uuu uuu.auto-imx6ull14x14evk

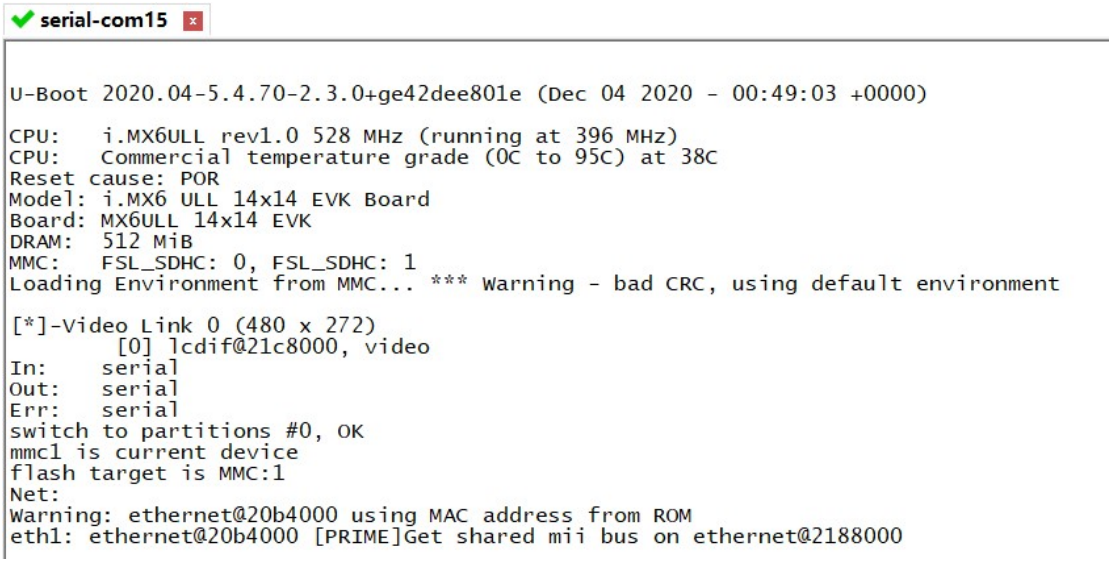
  Several minutes later, it will done.

[note]

During downloading images, one can start terminal software and connect debug UART, logs can be showed by the terminal software, for example, SecureCRT.

Step 5: Booting i.MX6ULL-EVK board

- Starting SecureCRT and connect COM15
- Power off board and Set SW602 to be internal boot
- Power on board, starting logs will be showed on SecureCRT



Input root to log in linux system:

```
NXP i.MX Release Distro 5.4-zeus imx6ul7d ttymxc0

imx6ul7d login: [   33.128415] VSD_3V3: disabling
[   33.131552] can-3v3: disabling

imx6ul7d login:
imx6ul7d login:
imx6ul7d login: root
Last login: Sat Nov 14 14:43:39 UTC 2020 on tty7
root@imx6ul7d:~#
root@imx6ul7d:~#
```

Step 6: Downloading 88W8801 Linux driver and cross compiling it.

- **Downloading 88W8801 Linux Driver**

https://www.nxp.com/products/wireless/wi-fi-plus-bluetooth/88w8801-2-4-ghz-single-band-1x1-wi-fi-4-802-11n-solution:88W8801?tab=Design_Tools_Tab

the wifi/bt driver source code requires NDA, so users can submit ticket to us in salesforce to Sign NDA, and support engineer will send drivers out.

Here we use *SD-UAPSTA-8801-U16-MMC-W14.68.36.p160-C4X14687_R0-MGPL*.zip

- **Decompressing Linux driver source code**

cd ~/

mkdir wifi-drivers

cd wifi-drivers

mkdir 88W8801

Copy *SD-UAPSTA-8801-U16-MMC-W14.68.36.p160-C4X14687_R0-MGPL*.zip to the directory.

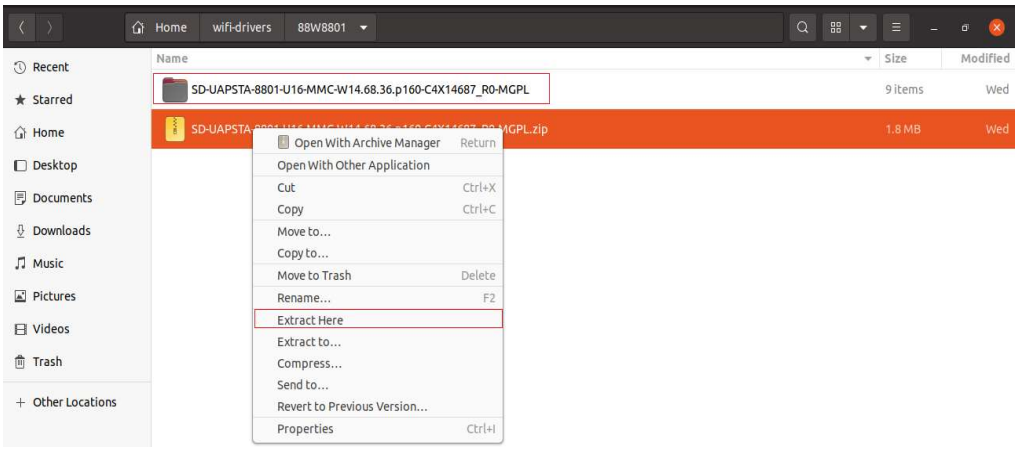
[Note]

If you downloaded the driver on windows, copy and paste it to Linux, if not, but ubuntu, use cp command to copy it.

cd 88W8801

cp ~/Downloads/*SD-UAPSTA-8801-U16-MMC-W14.68.36.p160-C4X14687_R0-MGPL*.zip ./

Decompress *SD-UAPSTA-8801-U16-MMC-W14.68.36.p160-C4X14687_R0-MGPL*.zip in file browser.



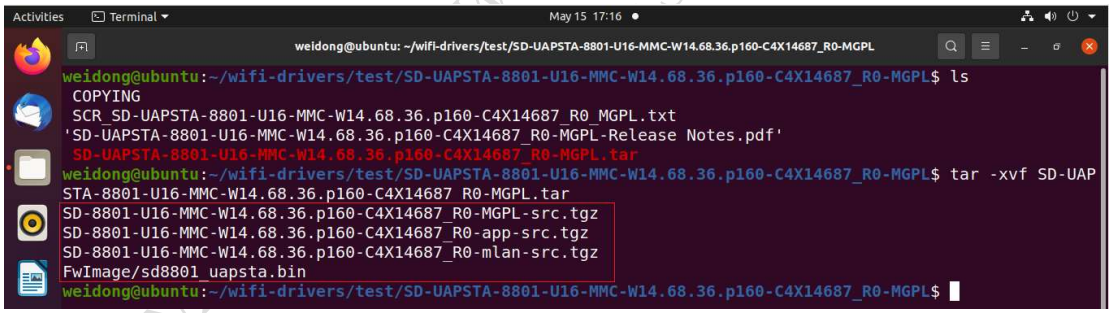
After decompressing it, a new directory is created in RED Rectangle.

In the directory, it's contents are like below:

Open terminal and enter the directory to decompress the tar ball.

tar -xvf SD-UAPSTA-8801-U16-MMC-W14.68.36.p160-C4X14687_R0-MGPL.tar

Then following files will be released.



Decompressing these 3 .tgz files:

tar xvf SD-8801-U16-MMC-W14.68.36.p160-C4X14687_R0-MGPL-src.tgz

tar xvf SD-8801-U16-MMC-W14.68.36.p160-C4X14687_R0-app-src.tgz

tar xvf SD-8801-U16-MMC-W14.68.36.p160-C4X14687_R0-mlan-src.tgz

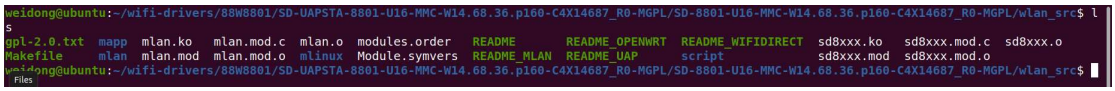
All source code has been decompressed into the directory marked in RED RECTANGLE.

Entering the directory

cd SD-8801-U16-MMC-W14.68.36.p160-C4X14687_R0-MGPL

cd wlan_src

Now we has entered into the directory of wifi driver source code.



Remember the directory:

~/wifi-drivers/88W8801/SD-UAPSTA-8801-U16-MMC-W14.68.36.p160-C4X14687_R0-MGPL/SD-8801-U16-MMC-W14.68.36.p160-C4X14687_R0-MGPL/wlan_src

- **Downloading the kernel source code of L5.4.70_2.3.0**

cd ~/

mkdir linux-source-code

cd linux-source-code

mkdir L5.4.70_2.3.0

cd L5.4.70_2.3.0

git clone https://source.codeaurora.org/external/imx/linux-imx -b imx_5.4.70_2.3.0

Then kernel source code will downloaded in ./linux-imx directory.

cd linux-imx

Setting cross compiler and Compiling linux kernel:

source /opt/fsl-imx-fb/5.4-zeus/environment-setup-cortexa9t2hf-neon-poky-linux-gnueabi

make imx_v7_defconfig

make

Wait until compilation is done.

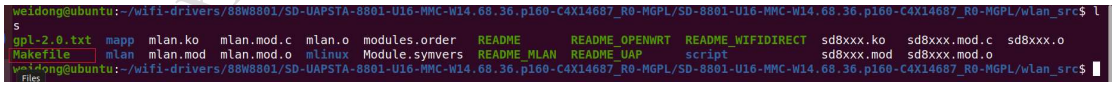
[Note]

Kernel directory is : /home/weidong/linux-source-code/L5.4.70_2.3.0/linux-imx

- Compiling 88W8801 Linux driver

cd ~/wifi-drivers/88W8801/SD-UAPSTA-8801-U16-MMC-W14.68.36.p160-C4X14687_R0-MGPL/SD-8801-U16-MMC-W14.68.36.p160-C4X14687_R0-MGPL/wlan_src

ls



One can modify KERNELDIR's value to be :

/home/weidong/linux-source-code/L5.4.70_2.3.0/linux-imx, like below:

```
119 #KERNELVERSION_X86 :=   $(shell uname -r)
120 #KERNELDIR ?= /lib/modules/$(KERNELVERSION_X86)/build
121 KERNELDIR ?= /home/weidong/linux-source-code/L5.4.70_2.3.0/linux-imx
122 LD += -S
```

make

[Note]

Users may not modify it, but need to specify the value KERNELDIR on the command line, like this:

make KERNELDIR=/home/weidong/linux-source-code/L5.4.70_2.3.0/linux-imx

Both methods can achieve the purpose of compiling WIFI driver.

After driver compilation is done, 2 .ko files are created in the wlan_src sub-director, see below, please!

```
MC-W14.68.36.p160-C4X14687_R0-MGPL/wlan_src$ ls
gpl-2.0.txt  mlan.ko     mlan.o          README          README_WIFIDIRECT  sd8xxx.mod.c
Makefile     mlan.mod    mlinux          README_MLAN     script             sd8xxx.mod.o
mapp         mlan.mod.c  modules.order   README_OPENWRT  sd8xxx.ko          sd8xxx.o
mlan         mlan.mod.o  Module.symvers  README_UAP      sd8xxx.mod
```

Step 7: Loading Linux driver of 88W8801 on i.MX6ULL-EVK

- **Copy firmware(sd8801_uapsta.bin) to iMX6ULL-EVK board**
  --copy the firmware from ubuntu to windows
  --Starting MobaXterm and login **iMX6ULL-EVK board by SSH via eth1 or eth0**
  --drag and drop **sd8801_uapsta.bin to /lib/firmware/nxp**
  If /lib/firmware/nxp/ doesn't exist, create it, please!
- **Copy mlan.ko and sd8xxx.ko to iMX6ULL-EVK board**

```
root@imx6ul/d:~# ls
mlan.ko  sd8801.ko  sd8801_uapsta.bin
root@imx6ul7d:~#
```

- **Checking if SDIO driver finds SDIO cards from booting logs**

root@imx6ul7d:~# dmesg | grep mmc
[0.000000] Kernel command line: console=ttymxc0,115200 root=/dev/mmcblk1p2 rootwait rw
[2.028929] mmc0: SDHCI controller on 2190000.usdhc [2190000.usdhc] using ADMA
[2.070078] mmc1: SDHCI controller on 2194000.usdhc [2194000.usdhc] using ADMA
[2.081394] mmc0: new high speed SDIO card at address 0001
[2.129574] mmc1: host does not support reading read-only switch, assuming write-enable
[2.144540] mmc1: new high speed SDHC card at address aaaa
[2.152901] mmcblk1: mmc1:aaaa SC16G 14.8 GiB
[2.178299] mmcblk1: p1 p2
[2.704011] EXT4-fs (mmcblk1p2): recovery complete
[2.710139] EXT4-fs (mmcblk1p2): mounted filesystem with ordered data mode. Opts: (null)
[6.087588] EXT4-fs (mmcblk1p2): re-mounted. Opts: (null)
root@imx6ul7d:~#

- **Loading 88W8801 driver**

root@imx6ul7d:~# insmod mlan.ko
[1183.966423] mlan: loading out-of-tree module taints kernel.
root@imx6ul7d:~# insmod sd8801.ko 'fw_name=nxp/sd8801_uapsta.bin'
[1337.535732] Wlan: FW download over, firmwarelen=260184 downloaded 260184
[1337.768475] fw_cap_info=0xba3, dev_cap_mask=0xffffffff

```
root@imx6ul7d:~# ifconfig -a
……
mlan0     Link encap:Ethernet  HWaddr d8:c0:a6:5c:09:71
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

uap0      Link encap:Ethernet  HWaddr d8:c0:a6:5c:0a:71
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

wfd0      Link encap:Ethernet  HWaddr da:c0:a6:5c:09:71
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
……
```


Step 8: Configuring 88W8801 and accessing Internet

```
root@imx6ul7d:~# ifconfig mlan0 up
root@imx6ul7d:~# wpa_passphrase NXPOPEN Use4Internet >> /etc/wpa_supplicant.conf
root@imx6ul7d:~# cat /etc/wpa_supplicant.conf
ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=0
update_config=1

network={
        key_mgmt=NONE
}
network={
        ssid="NXPOPEN"
        #psk="Use4Internet"
        psk=c5397a26ced00bcb3545de1e0b421f76c9b6ed2c72a36150b87d5951b755cf7c
}
root@imx6ul7d:~# ifconfig eth0 down
root@imx6ul7d:~# ifconfig eth1 down
root@imx6ul7d:~# wpa_supplicant -B -D wext -i mlan0 -c /etc/wpa_supplicant.conf
```

```
root@imx6ul7d:~# udhcpc -i mlan0
udhcpc: started, v1.31.0
udhcpc: sending discover
udhcpc: sending select for 192.168.1.230
udhcpc: lease of 192.168.1.230 obtained, lease time 14400
/etc/udhcpc.d/50default: Adding DNS 208.67.222.222
/etc/udhcpc.d/50default: Adding DNS 208.67.222.220
root@imx6ul7d:~# ifconfig
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:173 errors:0 dropped:0 overruns:0 frame:0
          TX packets:173 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:12482 (12.1 KiB)  TX bytes:12482 (12.1 KiB)

mlan0     Link encap:Ethernet  HWaddr d8:c0:a6:5c:09:71
          inet addr:192.168.1.230  Bcast:192.168.1.255  Mask:255.255.254.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:228 errors:0 dropped:0 overruns:0 frame:0
          TX packets:55 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:37088 (36.2 KiB)  TX bytes:5052 (4.9 KiB)

root@imx6ul7d:~# ping 114.114.114.114
PING 114.114.114.114 (114.114.114.114) 56(84) bytes of data.
64 bytes from 114.114.114.114: icmp_seq=1 ttl=80 time=39.6 ms
64 bytes from 114.114.114.114: icmp_seq=2 ttl=68 time=29.3 ms
64 bytes from 114.114.114.114: icmp_seq=3 ttl=94 time=44.1 ms
64 bytes from 114.114.114.114: icmp_seq=4 ttl=72 time=39.8 ms
64 bytes from 114.114.114.114: icmp_seq=5 ttl=72 time=38.0 ms
64 bytes from 114.114.114.114: icmp_seq=6 ttl=65 time=37.3 ms
64 bytes from 114.114.114.114: icmp_seq=7 ttl=87 time=66.1 ms
^C
--- 114.114.114.114 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6011ms
rtt min/avg/max/mdev = 29.284/42.038/66.149/10.676 ms
root@imx6ul7d:~#
```

[Need to pay attention to]

- **IO voltage between AW-NM191-uSD and i.MX6ULL-EVK uSDHC1(SD1)**
  - **On platform side**
    NVCC_SD is IO voltage for SD1, it has been designed as switchable one between 3.3V and 1.8V, the purpose is for high speed SD memory card. But for WIFI(SDIO card), the switching between 3.3V & 1.8V can not be implemented. By default, the IO voltage on platform side is 3.3V.
  - **On AW-NM191-uSD side**
    3.3V and 1.8V IO are both supported, but need to adjust it through jumper manually.

VIO_SD voltage level options

- For 3.3V supply, please connect J11 (2-3).
- For 1.8V supply, please connect J11 (1-2).

VIO voltage level options

- For 3.3V supply, please connect J4 (2-3).
- For 1.8V supply, please connect J4 (1-2).

For more detailed information, see the AW link, please!
http://www.azurewave.com/wireless-modules-nxp.html

AW-NM191MA	88W8801	Datasheet	下載
AW-NM191-uSD	88W8801	User's Guide	下載

- **Using 1.8V IO between Platform side and WIFI side**
  If you don't want to modify software, you can remove **Q703** on the bottom of SOM board, and short it's **pin1 and pin3**. Then NVCC_SD will be 1.8V.

Then set **AW-NM191-uSD IO voltage level to be 1.8V.**

- **About the limitation of SDIO clock speed**
  Due to hardware limitation, uSDHC clock speed is limited to Max 132MHz , we can get the result from kernel device tree, see below, please!

In arch/arm/boot/dts/imx6ull.dtsi :

```
&usdhc1 {
    compatible = "fsl,imx6ull-usdhc", "fsl,imx6sx-usdhc";
    assigned-clocks = <&clks IMX6UL_CLK_USDHC1_SEL>, <&clks IMX6UL_CLK_USDHC1>;
    assigned-clock-parents = <&clks IMX6UL_CLK_PLL2_PFD2>;
    assigned-clock-rates = <0>, <132000000>;
};

&usdhc2 {
    compatible = "fsl,imx6ull-usdhc", "fsl,imx6sx-usdhc";
    assigned-clocks = <&clks IMX6UL_CLK_USDHC2_SEL>, <&clks IMX6UL_CLK_USDHC2>;
    assigned-clock-parents = <&clks IMX6UL_CLK_PLL2_PFD2>;
    assigned-clock-rates = <0>, <132000000>;
};
```

So we can see 2 Ips CLOCK are both fixed max 132MHz.

Actually according to experience from some customers, Users can modify it to a maximum of 150MHz, if the frequency value is larger than 150MHz, it will not work.

NXP CAS-TIC Wireless MCU Team
Weidong Sun
05/16/2021