UM12366

Getting Started with NXP-based Wireless Modules on STM32MP257F-DK Running OpenST Linux OS

Rev. 1.0 — 8 September 2025 User manual

Document information

Information	Content
Keywords	host system, configuration, setup, pre-built image, wireless module, uSD-M.2 adapter, UART connection, firmware, driver, Wi-Fi bring-up, Bluetooth bring-up, Wi-Fi features, Bluetooth features
Abstract	Details the enabling of wireless solutions on STM32MP257F-DK.

1 About this document

The document details the enabling of wireless solutions on STM32MP257F-DK. The STM32MP257F-DK is powered by OpenSTLinux Distribution software and the NXP Linux drivers are used for NXP-based wireless modules. The manual covers:

- The bring-up of STM32MP257F-DK
- The configurations for the BSP image
- The hardware connection with NXP-based wireless modules
- The bring up of Wi-Fi and Bluetooth radios

1.1 Supported products

- IW416 (ref.[10])
- IW610 (ref.[11])
- IW612 (ref.[12])

2 STM32MP257F-DK overview

2.1 Discovery kit with STM32MP257F-DK MPU

The STM32MP257F-DK Discovery kit is designed as a complete demonstration and development platform for the STMicroelectronics STM32MP257FAK3 based on the Arm® Cortex®A35 and M33.

The product leverages the capabilities of STM32MP2 series microprocessors to allow users to develop applications using STM32 MPU OpenSTLinux Distribution software for the main processor (Arm® dual core Cortex®-A35) and STM32CubeMP2 software for the coprocessor (Arm® Cortex®-M33).

3 STM32MP257F-DK Linux image setup

3.1 Using the pre-build image

Steps to prepare eMMC to boot up STM32MP257F-DK kit:

Step 1 – Download the prebuild image (ref.[6]).

Step 2 – Flash the image, install STM32CubeProgrammer tool. See ref.[7].

3.2 Booting from eMMC

To boot the STM32MP257F-DK from eMMC, set the boot switch as detailed in ref.[8].

This guide is based on the version: *en.FLASH-stm32mp2-openstlinux-6.6-yocto-scarthgap-mpu-v24.11.06* (ref.[3]).

3.2.1 Serial console setup

Steps to setup the serial console and access the STM32MP257F-DK device terminal:

- Open the serial console and log into the device.

```
ubuntu@ubuntu-desktop:/# sudo minicom -D /dev/ttySTMx
```

In the command, `ttySTM1` are the serial devices. The minicom setup configuration is as follows:

```
A -  Serial Device     :  /dev/ttySTMx
E -  Bps/Par/Bits    :  115200 8N1
F -  Hardware Flow Control    :  No
G -  Software Flow Control    :  No
```

- Save and exit.

3.2.2 Linux OS login

The default login username for the openST Linux OS is *root*. There is no password.

3.2.3 Network connectivity

To interact with STM32MP257F-DK and transfer files, connect the STM32MP257F-DK Discovery Kit to the same network as the development system.

On the STM32MP257F-DK, use a terminal window to retrieve the network configuration for the on-board ethernet port.

```
ip addr show end0
```

Example of command output:

```
end0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen
 1000
    link/ether 10:e7:7a:e3:d2:9e brd ff:ff:ff:ff:ff:ff
    altname ethernet0
    altname ethernet0.eth1
    inet 172.29.122.140/24 metric 10 brd 172.29.122.255 scope global dynamic end0
    valid_lft 522sec preferred_lft 522sec
    inet6 fe80::12e7:7aff:fee3:d29e/64 scope link proto kernel_ll
    valid_lft forever preferred_lft forever
```

4 Host system configuration

4.1 Install additional packages

Steps to install the required additional packages on the host system:

Step 1 – Update the package index on the host system..

```
apt update
```

Step 2 – Install the additional packages.

```
apt install dtc nano git
```

4.2 Enable the peripheral interface

Steps to enable UART interface on the 40-pin expansion header (ref.[9]):

Step 1 – Go to the directory with the device tree.

```
cd  /boot/
```

Step 2 – Decompile the currently used device tree file.(dtb file into dts file)

```
dtc -I dtb -O dts stm32mp257f-dk.dtb -o tree.dts
```

Step 3 – Open the decompiled device tree with the nano editor.

```
nano tree.dts
```

Step 4 – Locate the node named `serial@40220000`. Change the parameter `status = “okay”`.

Step 5 – Compile the updated device tree source (.dts) file.

```
dtc -I dts -O dtb tree.dts -o stm32mp257f-dk.dtb
```

Step 6 – Reboot the board to load compiled device tree binary (.dtb) file.

```
reboot
```

5 Host system setup

5.1 Set up the development environment

Follow the guide to setup the development environment on the host development system. See ref.[4].

- Download the SDK
- Run the installation script
- Start the SDK

Note: *You can skip the step to create a simple application. Rebuilding the Linux kernel is not required.*

5.2 Set up the Linux kernel build environment

Follow the guide to modify, rebuild, and reload the Linux kernel. See ref.[5].

- Download the BSP with the Linux kernel source code.
- Prepare the Linux kernel source code.

The Linux kernel does not have to be built, but it must be prepared.

Command to prepare the kernel in the Linux kernel directory:

```
make O=${OUTPUT_BUILD_DIR} prepare
```

Versions of Linux kernel used:

en.SDK-x86_64-stm32mp2-openstlinux-6.6-yocto-scarthgap-mpu-v24.11.06.tar.gz

en.SOURCES-stm32mp2-openstlinux-6.6-yocto-scarthgap-mpu-v24.11.06.tar.gz

6 Host system software installation

Requirement: bring up the host development system as explained in Section 5.

6.1 Compile the kernel modules with RFCOMM support

Step 1 – Open your kernel config.

```
cd $HOME/STM32MPU_workspace/STM32MPU-Ecosystem-v6.0.0/ Developer-Package/stm32mp2-
openstlinux-6.6-yocto-scarthgap-mpu-v24.11.06/
sources/aarch64-ostl-linux/linux-stm32mp-6.6.48-stm32mp-r1-r0/
linux-6.6.48
```

Step 2 – In kernel build directory, run the menuconfig.

```
make menuconfig
```

Step 3 – Enable RFCOMM support.

Within the menuconfig interface, navigate through the following menu path:

```
Networking support  —>
   Bluetooth subsystem support  --->
    [M] RFCOMM protocol support
    [*]   RFCOMM TTY support
```

Where:

- `[M]` selects the RFCOMM protocol support as a kernel module.
- `[*]` enables RFCOMM TTY support, which allows serial communication over Bluetooth.

Step 4 – Save and exit the menuconfig.

Step 5 – Compile kernel driver modules.

```
make O=${OUTPUT_BUILD_DIR} modules
```

Step 6 – Copy the RFCOMM driver module to STM32MP257F-DK.

```
scp {OUTPUT_BUILD_DIR}/install_artifact/lib/modules/6.6.48/kernal/
net/bluetooth/rfcomm/rfcomm.ko root@<ip>:/lib/modules/6.6.48/extra/
```

6.2 Compile NXP Wi-Fi driver

Steps to download NXP Wi-Fi driver, compile and transfer the files to STM32MP257F-DK:

Step 1 – Go to the base of your working environment.

```
cd $HOME/STM32MPU_workspace/STM32MPU-Ecosystem-v6.0.0
```

Step 2 – Clone the driver repository from NXP GitHub project.

```
git clone https://github.com/nxp-imx/mwifiex.git
cd mwifiex/
```

Step 3 – Enable the support for NXP Wi-Fi device.

Use the nano editor to open and edit the Makefile.

```
nano Makefile
```

Enable (set to `y`) the configuration flag for the Wi-Fi device. Look for the following lines in the Makefile and edit them as needed:

```
# Multi-chipsets
CONFIG_SD8978=y     # Enables IW416
CONFIG_SD9177=y     # Enables IW612
CONFIG_SDIW610=y    # Enables IW610
```

If a line is missing or if the value is set to `n`, add the line or change the value to `y`.

Example of Makefile content:

```
# Multi-chipsets
CONFIG_SD8978=y
CONFIG_SD8897=n
[...]
CONFIG_SD8997=y
CONFIG_USB8997=n
CONFIG_PCIE8997=y
CONFIG_SD8987=y
CONFIG_SD9097=n
CONFIG_SD9177=y
[....]
CONFIG_SDIW610=y
CONFIG_USBIW610=n
```

Step 4 – Compile NXP Wi-Fi driver.

```
make O="${OUTPUT_BUILD_DIR}"  KERNELDIR=~/STM32MPU_workspace/
STM32MPU-Ecosystem-v6.0.0/Developer-Package/stm32mp2-openstlinux-6.6-yocto-scarthgap-
mpu-v24.11.06/sources/aarch64-ostl-linux/linux-stm32mp-6.6.48-stm32mp-r1-r0/linux-6.6.48
 build
```

Step 5 – Copy the NXP Wi-Fi driver from your host system to the e STM32MP257F-DK in the */lib/modules/6.6.48/extra/* directory.

Create the directory on the STM32MP257F-DK if it does not already exist.

```
scp mlan.ko moal.ko root@<ip>:/lib/modules/6.6.48/extra/
```

6.3 Install NXP wireless firmware

Download and install the firmware for NXP wireless modules:

Step 1 – Create the destination directory for the firmware and the configuration file with the parameters for the driver load to STM32MP257F-DK.

```
mkdir -p /lib/firmware/nxp/
```

Step 2 – Clone the firmware repository from NXP GitHub space.

```
cd ~
git clone https://github.com/nxp-imx/imx-firmware.git
git checkout if-6.12.3_1.0.0
cd imx-firmware/
```

Step 3 – Copy the firmware to the appropriate firmware directory.

Example of command for IW612:

```
cp nxp/FwImage_IW612_SD/sduart_nw61x_v1.bin.se /lib/firmware/nxp/
```

Step 4 – Copy the configuration file with the parameters for the driver load (*wifi_mod_para.conf*) to the firmware directory.

```
cp nxp/wifi_mod_para.conf /lib/firmware/nxp/
```

Note: *On the STM32MP257F-DK board, USART6 is used by the UCSI firmware that runs on the Cortex-M33 core. Both the Cortex-A35 (Linux) and Cortex-M33 (firmware) may attempt to access USART6 and its associated clock resources simultaneously. To avoid resource conflicts related to USART6, the recommendation is to disable or stop the firmware service that manages the M33 firmware.*

Step 5 – Stop or disable the firmware services.

Command to stop the firmware services:

```
systemctl stop st-m33firmware-load.service
```

Command to disable the firmware services:

```
systemctl disable st-m33firmware-load.service
```

Step 6 – Reboot the host system for the changes to take effect.

```
reboot
```

Command output:

```
SYSTEM REBOOT
```

7 NXP module setup with STM32MP257F-DK

To enable the Wi-Fi interface, connect the uSD-M.2 adapter equipped with NXP Wi-Fi module to the SD card slot (CN6) of the STM32MP257F-DK board.

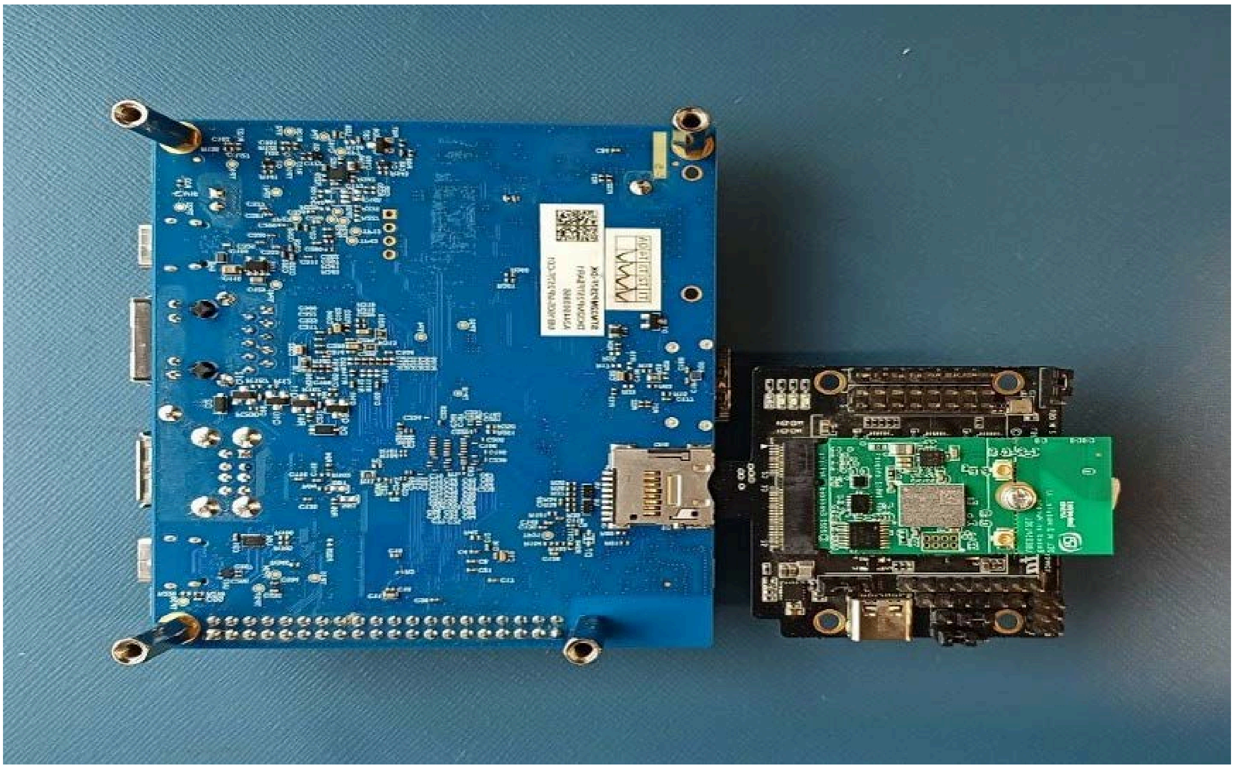

Figure 1. Connecting NXP Wi-Fi module to STM32MP257F-DK board

For Bluetooth communication, use the UART interface exposed on the 40-pin expansion header of the STM32MP257F-DK board.

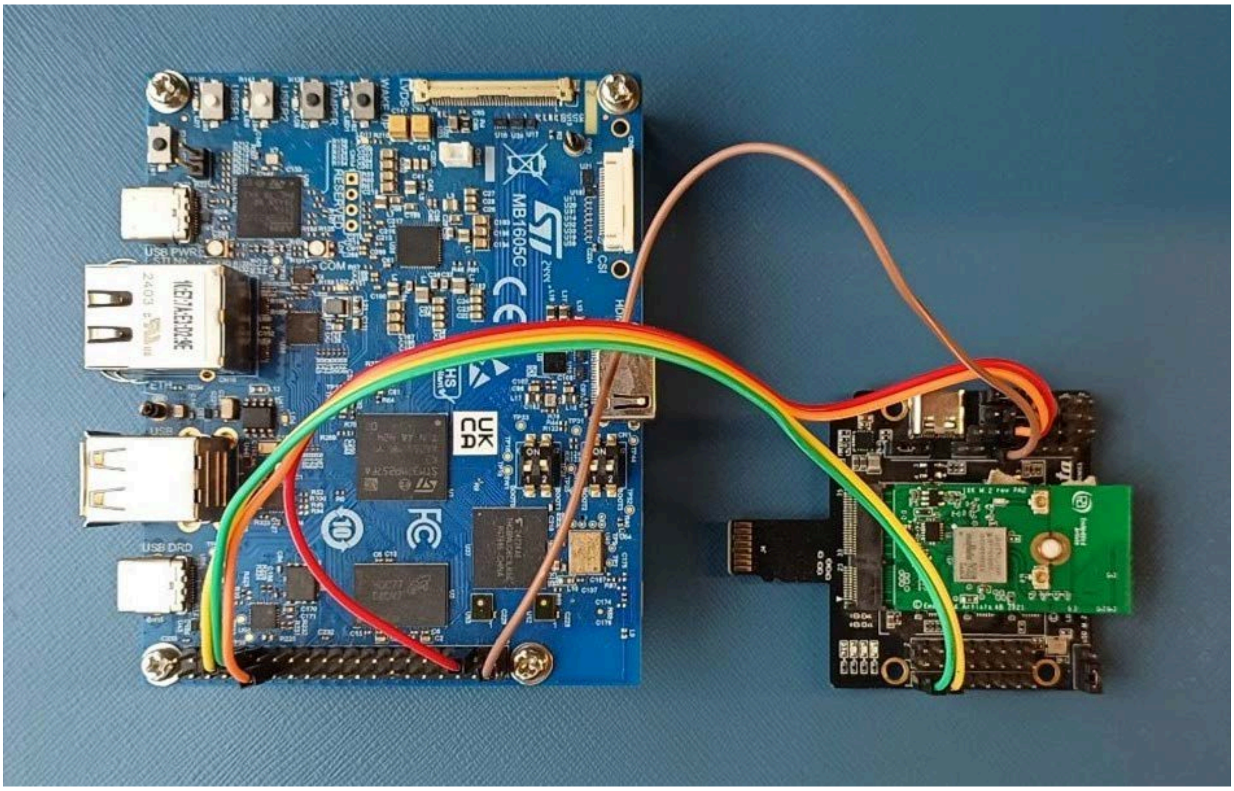

Figure 2. Connection of UART peripherals on NXP module and STM32MP257F-DK board

Table 1 includes the pin assignment and signal mapping for the Bluetooth module connection.

Table 1. UART signals on NXP module and STM32MP257F-DK

uSD-M.2 adapter	STM32MP257F-DK (Pin of 40-pin GPIO header)
J9 – Pin 2 (BT_UART_RXD_HOST)	Pin 8 (PF13) (USART6_TX)
J9 – Pin 1 (BT_UART_TXD_HOST)	Pin 10 (PF14) (USART6_RX)
J8 – Pin 4 (BT_UART_CTS_HOST)	Pin 11 (PG5) (USART6_RTS)
J8 – Pin 3 (BT_UART_RTS_HOST)	Pin 36 (PF15) (USART6_CTS)
J7 – Pin 6 (GND)	Pin 39 (GND)

7.1 IW416-based Murata module LBEE5CJ1XK

See the section *NXP-based wireless modules* in ref.[1].

7.2 IW610-based Murata module LBES0ZZ2LL

See the section *NXP-based wireless modules* in ref.[1].

7.3 IW612-based Murata module LBES5PL2EL

See the section *NXP-based wireless modules* in ref.[1].

8 Bring-up of Wi-Fi

The section describes the bring-up of the Wi-Fi interface of NXP-based wireless modules connected to STM32MP257F-DK board.

8.1 Bring-up of IW416 wireless module

Step 1 – Use an external power supply for IW416 module (recommendation).

To power the IW416 module externally, move the J1 jumper to the position 1-2. J1 is the jumper used for power supply selection The setting enables a 5 V/3.3 V VBAT supply from the micro-USB port (J2) of the adapter.

Steps to update the DTB File for IW416 module:

Step 1 – Go to the directory with the device tree.

```
cd /boot/
```

Step 1 – Decompile the device tree file in use (dtb file into dts file).

```
dtc -I dtb -O dts stm32mp257f-dk.dtb -o tree.dts
```

Step 2 – Open the decompiled device tree with the nano editor.

```
nano tree.dts
```

Step 3 – Locate the node named `mmc@4822000`. Disable some configurations.

```
#sd-uhs-sdr12;
#sd-uhs-sdr25;
#sd-uhs-sdr50;
#sd-uhs-ddr50;
#sd-uhs-sdr104;
```

Step 4 – Compile the updated device tree source (.dts) file.

```
dtc -I dts -O dtb tree.dts -o stm32mp257f-dk.dtb
```

Step 5 – Reboot the board to load compiled device tree binary (.dtb) file.

```
reboot
```

Steps to load the driver and bring up the IW416-based wireless module:

Step 1 – Use the nano editor to verify the module parameters in *wifi_mod_para.conf* configuration file

```
root@stm32mp2-e3-d2-9e:~# nano /lib/firmware/nxp/wifi_mod_para.conf
```

Content of the configuration file

```
SDIW416 = {
        cfg80211_wext=0xf
        max_vir_bss=1
        cal_data_cfg=none
        ps_mode=1
        auto_ds=1
        host_mlme=1
        fw_name=nxp/sduartiw416_combo.bin
}
```

Step 2 – Verify that the Linux kernel module dependency database is up-to-date.

```
root@stm32mp2-e3-d2-9e:~# depmod -a
```

Step 3 – Load the modules into the Linux kernel.

```
root@stm32mp2-e3-d2-9e:~#modprobe moal mod_para=nxp/wifi_mod_para.conf
```

Step 4 – Verify the kernel debug messages in the command output.

```
root@stm32mp2-e3-d2-9e:~#modprobe moal mod_para=nxp/wifi_mod_para.conf
[   48.122603] wlan: Loading MWLAN driver
[   48.123213] wlan: Register to Bus Driver...
[   48.125263] vendor=0x02DF device=0x9159 class=0 function=1
[   48.130588] Attach moal handle ops, card interface type: 0x108
[   48.136361] rps set to 0 from module param
[   48.142241] SDIW416: init module param from usr cfg
[   48.145357] card_type: SDIW416, config block: 0
[   48.149804] cfg80211_wext=0xf
[   48.152753] max_vir_bss=1
[   48.155393] cal_data_cfg=none
[   48.158330] ps_mode = 1
[   48.160748] auto_ds = 1
[   48.163286] host_mlme=enable
[   48.166108] fw_name=nxp/sduartiw416_combo.bin
[   48.170474] SDIO: max_segs=341 max_seg_size=131008
[   48.175329] rx_work=1 cpu_num=2
[   48.178369] Enable moal_recv_amsdu_packet
[   48.182434] Attach mlan adapter operations.card_type is 0x108.
[   48.188848] wlan: Enable TX SG mode
[   48.191803] wlan: Enable RX SG mode
[   48.198399] Request firmware: nxp/sduartiw416_combo.bin
[   48.442537] Wlan: FW download over, firmwarelen=400628 downloaded 391908
[   50.174446] WLAN FW is active
[   50.174480] on_time is 49904294500
[   50.214591] VDLL image: len=8720
[   50.218438] FW country code WW does not match with US
[   50.222405] fw_cap_info=0x187ccf03, dev_cap_mask=0xffffffff
[   50.223446] max_p2p_conn = 8, max_sta_conn = 8
[   50.366604] Register NXP 802.11 Adapter mlan0
[   50.380068] Register NXP 802.11 Adapter uap0
[   50.406548] Register NXP 802.11 Adapter wfd0
[   50.406620] wlan: version = SDIW416---16.92.21.p149.2-MM6X16505.p14-GPL-(FP92)
[   50.420402] wlan: Register to Bus Driver Done
[   50.420438] wlan: Driver loaded successfully
```

Step 5 – Verify the Wi-Fi interfaces

```
root@stm32mp2-e3-d2-9e:~# ifconfig -a
```

Command output example:

```
end0      Link encap:Ethernet  HWaddr 10:E7:7A:E3:D2:9E
          inet addr:172.29.122.140 Bcast:172.29.122.255
          Mask:255.255.255.0
          inet6 addr: fe80::12e7:7aff:fee3:d29e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:113 errors:0 dropped:38 overruns:0 frame:0
          TX packets:73 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:11451 (11.1 KiB)  TX bytes:10949 (10.6 KiB)
          Interrupt:59 Base address:0x8000
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:90 errors:0 dropped:0 overruns:0 frame:0
          TX packets:90 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:7107 (6.9 KiB)  TX bytes:7107 (6.9 KiB)
mlan0   Link encap:Ethernet  HWaddr 9C:50:D1:45:37:09
          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 9E:50:D1:45:38:09
          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)
```

8.2 Bring-up IW610 wireless module

Steps to update the *.dtb* File for IW610 module:

Step 1 – Navigate to the directory that contains the device tree.

```
cd /boot/
```

Step 2 – Decompile the currently used device tree file (*.dtb* file into *.dts* file).

```
dtc -I dtb -O dts stm32mp257f-dk.dtb -o tree.dts
```

Step 3 – Open the decompiled device tree with the nano editor.

```
nano tree.dts
```

Step 4 – Locate the node named `mmc@48220000`. Change the parameter `max-frequency = <0x17D7840>;`.

Step 5 – Compile the updated device tree source (.dts) file.

```
dtc -I dts -O dtb tree.dts -o stm32mp257f-dk.dtb
```

Step 6 – Reboot the board to load compiled device tree binary (.dtb) file.

```
reboot
```

Steps to load the driver modules and bring up the IW610-based wireless module.

Step 1 – Use the nano editor to verify the module parameters in wifi_mod_para.conf configuration file.

```
root@stm32mp2-e3-d2-9e:~#nano /lib/firmware/nxp/wifi_mod_para.conf
```

Content of the configuration file:

```
SDIW610 = {
        cfg80211_wext=0xf
        max_vir_bss=1
        cal_data_cfg=none
        ps_mode=1
        auto_ds=1
        host_mlme=1
        fw_name=nxp/sduart_iw610.bin.se
}
```

Step 2 – Verify that the Linux kernel module dependency database is up-to-date.

```
root@stm32mp2-e3-d2-9e:~# depmod -a
```

Step 3 – Load the modules into the Linux kernel.

```
root@stm32mp2-e3-d2-9e:~#modprobe moal mod_para=nxp/wifi_mod_para.conf
```

Step 4 – Verify the kernel debug messages in the command output.

```
root@stm32mp2-e3-d2-9e:~#modprobe moal mod_para=nxp/wifi_mod_para.conf
[  723.185385] wlan: Loading MWLAN driver
[  723.186011] wlan: Register to Bus Driver...
[  723.188199] vendor=0x0471 device=0x0215 class=0 function=1
[  723.193692] Attach moal handle ops, card interface type: 0x10d
[  723.199187] rps set to 0 from module param
[  723.203528] SDIW610: init module param from usr cfg
[  723.208184] card_type: SDIW610, config block: 0
[  723.212640] cfg80211_wext=0xf
[  723.215579] max_vir_bss=1
[  723.218237] cal_data_cfg=none
[  723.221159] ps_mode = 1
[  723.223694] auto_ds = 1
[  723.226135] host_mlme=enable
[  723.228958] fw_name=nxp/sduart_iw610.bin.se
[  723.233223] SDIO: max_segs=341 max_seg_size=131008
[  723.237973] rx_work=1 cpu_num=2
[  723.241120] Enable moal_recv_amsdu_packet
[  723.245186] Attach mlan adapter operations.card_type is 0x10d.
[  723.251558] wlan: Enable TX SG mode
[  723.254511] wlan: Enable RX SG mode
[  723.262384] Request firmware: nxp/sduart_iw610.bin.se
[  725.043597] Wlan: FW download over, firmwarelen=828548 downloaded 812948
[  725.490270] WLAN FW is active
[  725.490304] on_time is 725220275975
[  725.522395] VDLL image: len=15600
[  725.530378] fw_cap_info=0x487cbf03, dev_cap_mask=0xffffffff
[  725.530425] uuid: 83d861b492165a6696476e79abdc3947
[  725.535189] max_p2p_conn = 8, max_sta_conn = 10
[  725.706472] Register NXP 802.11 Adapter mlan0
[  725.706639] wlan: uap%d set max_mtu 2000
[  725.719671] Register NXP 802.11 Adapter uap0
[  725.818446] Register NXP 802.11 Adapter wfd0
[  725.818518] wlan: version = SDIW610---18.99.5.p51-MM6X18505.p14-GPL-(FP92)
[  725.834293] Set REG 0x45001064: 0xc000 slew_rate=3
[  725.851962] wlan: Register to Bus Driver Done
[  725.851994] wlan: Driver loaded successfully
```

Step 5 – Verify the Wi-Fi interfaces.

```
root@stm32mp2-e3-d2-9e:~# ifconfig -a
```

Command output example:

```
end0      Link encap:Ethernet  HWaddr 10:E7:7A:E3:D2:9E
          inet addr:172.29.122.140 Bcast:172.29.122.255
          Mask:255.255.255.0
          inet6 addr: fe80::12e7:7aff:fee3:d29e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:4127 errors:0 dropped:112 overruns:0 frame:0
          TX packets:773 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5103798 (4.8 MiB)  TX bytes:90789 (88.6 KiB)
          Interrupt:59
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:90 errors:0 dropped:0 overruns:0 frame:0
          TX packets:90 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:7107 (6.9 KiB)  TX bytes:7107 (6.9 KiB)
mlan0   Link encap:Ethernet  HWaddr 78:F5:05:7B:CD:30
          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 7A:F5:05:7B:CE:30
          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)
```

8.3 Bring-up of IW612 wireless module

Steps to load the driver module and bring up the IW612-based wireless module:

Step 1 – Use the nano editor to verify the module parameters in *wifi_mod_para.conf* configuration file.

```
root@stm32mp2-e3-d2-9e:~# nano /lib/firmware/nxp/wifi_mod_para.conf
```

Content of the configuration file

```
SDIW612 = {
        cfg80211_wext=0xf
        max_vir_bss=1
        cal_data_cfg=none
        ps_mode=1
        auto_ds=1
        host_mlme=1
        fw_name=nxp/sduart_nw61x_v1.bin.se
}
```

Step 2 – Verify that the Linux kernel module dependency database is up-to-date.

```
root@stm32mp2-e3-d2-9e:~# depmod -a
```

Step 3 – Load the modules into the Linux kernel.

```
root@stm32mp2-e3-d2-9e:~# modprobe moal mod_para=nxp/wifi_mod_para.conf
```

Step 4 – Verify the kernel debug messages in the command output

```
[ 5416.168344] wlan: Loading MWLAN driver
[ 5416.169000] wlan: Register to Bus Driver...
[ 5416.172442] vendor=0x0471 device=0x0205 class=0 function=1
[ 5416.176407] Attach moal handle ops, card interface type: 0x109
[ 5416.182076] rps set to 0 from module param
[ 5416.186458] SDIW612: init module param from usr cfg
[ 5416.191093] card_type: SDIW612, config block: 0
[ 5416.195544] cfg80211_wext=0xf
[ 5416.198466] max_vir_bss=1
[ 5416.201104] cal_data_cfg=none
[ 5416.204040] ps_mode = 1
[ 5416.206460] auto_ds = 1
[ 5416.208898] host_mlme=enable
[ 5416.211836] fw_name=nxp/sduart_nw61x_v1.bin.se
[ 5416.216303] SDIO: max_segs=341 max_seg_size=131008
[ 5416.221049] rx_work=1 cpu_num=2
[ 5416.224189] Enable moal_recv_amsdu_packet
[ 5416.228152] Attach mlan adapter operations.card_type is 0x109.
[ 5416.234390] wlan: Enable TX SG mode
[ 5416.237551] wlan: Enable RX SG mode
[ 5416.242808] Request firmware: nxp/sduart_nw61x_v1.bin.se
[ 5416.608518] Wlan: FW download over, firmwarelen=1074212 downloaded 961412
[ 5417.026910] WLAN FW is active
[ 5417.026946] on_time is 5416756354800
[ 5417.059573] VDLL image: len=112800
[ 5417.066728] fw_cap_info=0x487cff03, dev_cap_mask=0xffffffff
[ 5417.066771] uuid: 98a828df5f68537b860cfea4c72a00f7
[ 5417.071538] max_p2p_conn = 8, max_sta_conn = 16
[ 5417.339110] Register NXP 802.11 Adapter mlan0
[ 5417.339263] wlan: uap%d set max_mtu 2000
[ 5417.354929] Register NXP 802.11 Adapter uap0
[ 5417.447083] Register NXP 802.11 Adapter wfd0
[ 5417.447169] wlan: version = SDIW612---18.99.3.p23.6-MM6X18437.p21-GPL-(FP92)
[ 5417.465112] wlan: Register to Bus Driver Done
[ 5417.465146] wlan: Driver loaded successfully
```

Step 5 – Verify the Wi-Fi interfaces.

```
root@stm32mp2-e3-d2-9e:~# ifconfig -a
```

Command output example:

```
end0      Link encap:Ethernet  HWaddr 10:E7:7A:E3:D2:9E
        inet addr:172.29.122.140  Bcast:172.29.122.255  Mask:255.255.255.0
        inet6 addr: fe80::12e7:7aff:fee3:d29e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:13010 errors:0 dropped:2509 overruns:0 frame:0
          TX packets:1564 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:6588657 (6.2 MiB)  TX bytes:235829 (230.3 KiB)
          Interrupt:59 Base address:0x4000
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:172 errors:0 dropped:0 overruns:0 frame:0
          TX packets:172 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:13317 (13.0 KiB)  TX bytes:13317 (13.0 KiB)
mlan0   Link encap:Ethernet  HWaddr 50:26:EF:A2:F7:5C
          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 52:26:EF:A2:F8:5C
          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)
```

9 Wi-Fi features and configurations

Refer to the section *Connectivity features* in ref.[2].

10 Bring-up of Bluetooth

The section describes the bring-up the Bluetooth interfaces for NXP wireless modules.

Step 1 – Open uart port.

```
exec 3<> /dev/ttySTM1
stty -F /dev/ttySTM1 115200
```

Note: *The Wi-Fi driver must be installed before NXP Bluetooth UART driver.*

Step 2 – Load the Wi-Fi driver with the combo firmware. Initialize the Wi-Fi interface (Section 8).

Step 3 – Load NXP Bluetooth UART driver.

```
modprobe btnxpuart
```

Step 4 – Attach ttySTM1 port.

```
hciattach /dev/ttySTM1 any 3000000 flow
```

Step 5 – Verify the status of *hci* interface.

```
root@stm32mp2-e3-d2-9e:~# hciconfig hci1 up
root@stm32mp2-e3-d2-9e:~# hciconfig -a
```

Command output example:

```
root@stm32mp2-e3-d2-9e:~# hciconfig -a
hci1:   Type: Primary  Bus: UART
        BD Address: 50:26:EF:A2:F7:5D  ACL MTU: 1021:7  SCO MTU: 120:6
        UP RUNNING
        RX bytes:1544 acl:0 sco:0 events:95 errors:0
        TX bytes:1287 acl:0 sco:0 commands:95 errors:0
        Features: 0xff 0xfe 0x8f 0xfe 0xdb 0xff 0x7b 0x87
        Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3
        Link policy: RSWITCH HOLD SNIFF
        Link mode: PERIPHERAL ACCEPT
        Name: 'stm32mp2-e3-d2-9e #2'
        Class: 0x000000
        Service Classes: Unspecified
        Device Class: Miscellaneous,
        HCI Version: 5.4 (0xd)  Revision: 0x8300
        LMP Version: 5.4 (0xd)  Subversion: 0x1017
        Manufacturer: NXP Semiconductors (formerly Philips Semiconductors)
```

10.1 Bluetooth setup instructions

Steps to enable experimental features in the Bluetooth service for advanced Bluetooth functionality on the STM32MP257F-DK:

Step 1 – Create the Bluetooth service override directory.

```
root@stm32mp2-e3-d2-9e:~# mkdir -p /etc/systemd/system/bluetooth.service.d
```

Step 2 – Create the override configuration file.

```
root@stm32mp2-e3-d2-9e:~#nano /etc/systemd/system/bluetooth.service.d/
override.conf
```

Step 3 – Add the following content to the file.

```
[Service]
ExecStart=
ExecStart=/usr/libexec/bluetooth/bluetoothd –experimental
```

Step 4 – Save and exit.

Step 5 – Reload the system configuration and restart the Bluetooth service.

```
root@stm32mp2-e3-d2-9e:~# systemctl daemon-reexec
root@stm32mp2-e3-d2-9e:~# systemctl daemon-reload
root@stm32mp2-e3-d2-9e:~# systemctl restart bluetooth.service
```

11 Bluetooth classic/Bluetooth LE features and configurations

Refer to the section *Connectivity features* in ref.[2].

12 References

[1] User manual – UM11483: Getting Started with NXP-based Wireless Modules on i.MX 8M Quad EVK Running Linux OS (link)

[2] Reference manual – RM00297: Linux Software Reference Manual for NXP Wireless Connectivity (link)

[3] GitHub – STMicroelectronics/meta-st-stm32mp – Flash version of STM32MP2 openSTLinux (*en.FLASH-stm32mp2-openstlinux-6.6-yocto-scarthgap-mpu-v24.11.06)* (link)

[4] Webpage – Develop on Arm® Cortex®-A35 (link)

[5] Webpage – Modify, rebuild and reload the Linux® kernel (link)

[6] Webpage – STM32MP25 Discovery Kit – Downloading the image and flashing it to the board (link)

[7] Webpage – STM32MP257x-DKx hardware description – Installing the STM32CubeProgrammer tool (link)

[8] Webpage – STM32MP257x-DKx hardware description – Booting switch (link)

[9] Webpage – STM32MP257x-DKx hardware description – GPIO expansion connector (link)

[10] Webpage – IW416: 2.4/5 GHz dual-band 1x1 Wi-Fi® 4 (802.11n) + Bluetooth® Solution (link)

[11] Webpage – IW610: 2.4/5GHz dual-band 1x1 Wi-Fi® 6 + Bluetooth Low Energy + 802.15.4 Tri-Radio Solution (link)

[12] Webpage – IW612: 2.4/5 GHz dual-band 1x1 Wi-Fi® 6 (802.11ax) + Bluetooth® + 802.15.4 Tri-radio Solution (link)

13 Note about the source code in the document

The example code shown in this document has the following copyright and BSD-3-Clause license:

14 Revision history

Table 2. Revision history

Document ID	Release date	Description
UM12366 v.1.0	8 September 2025	• Initial version

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, μVision, Versatile — are trademarks and/or registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved.

Bluetooth — the Bluetooth wordmark and logos are registered trademarks owned by Bluetooth SIG, Inc. and any use of such marks by NXP Semiconductors is under license.

Tables

Figures

Contents