Wi-Fi Setup on TRS Board (Zynq Processor)

Table of Contents

1) Download Wi-Fi Driver and Firmware Files

Download the *mwifiex* driver source code from the following GitHub repository: https://github.com/nxp-imx/mwifiex

This repository contains two directories: *mlan* and *mlinux*. These folders contain all required *.c* and *.h* files for the driver.

Download the *mwifiex* firmware from the following GitHub repository: https://github.com/nxp-imx/imx-firmware/tree/lf-6.12.3_1.0.0/nxp/FwImage_IW612_SD

Download the *'sduart_nw61x_v1.bin.se'* firmware from the link above.

2) Compiling the mwifiex Driver in PetaLinux

- Create a new *PetaLinux* project with a preferred name. You can follow the instructions provided in the *PetaLinux User Guide* for project creation.
- After creating the project, generate a new module named *mwifiex* using the following command:
  - ***petalinux-create -t modules --name mwifiex --enable***
- This command will create a new directory named mwifiex under the following path:
  - *"**project-spec/meta-user/recipes-modules/mwifiex/**"*
- Inside the mwifiex directory, you will find a *.bb file* and a files folder. The files folder contains a default *Makefile* and *a .c file*. Delete both the files.
- Now, copy the *mlan* and *mlinux* directories (downloaded from GitHub) into the files folder. Also, copy the appropriate Makefile required to build the driver.
- Next, edit the *.bb file* in the *mwifiex* directory and include all your source and header files in the recipe.
- Update the *.bb* recipe to set required *CFLAGS*, compile *mlan.ko* and *moal.ko* modules, and install them into the kernel's extra modules directory.
- To load the driver statically, add the following line to the *.bb* recipe:
  "KERNEL_MODULE_AUTOLOAD += "moal mlan"
- If this line is not added to the .bb recipe, then follow Step 5 to load the driver manually at runtime.

2.1) Adding Regulatory Database

- Download the following files using your web browser:
    - *regulatory.db*
    - *regulatory.db.p7s*
- Create a new directory under the following path:
    - "**project-spec/meta-user/recipes-bsp/<recipe-name>/files/"**
- Replace <*recipe-name*> with the name you want to give your recipe (for example, *regulatory-db*)
- Place both *regulatory.db* and *regulatory.db.p7s* into this file's directory.
- Create the *BitBake* recipe file in <*recipe-name*> directory:
    - **vi <recipe name>**
    - **Example:** *vi regualatory-db.bb*
- Paste the following content into the *BitBake* recipe file.

```
DESCRIPTION = "Install custom regulatory.db and regulatory.db.p7s"
LICENSE = "CLOSED"
PR = "r0"

SRC_URI += "file://regulatory.db \
file://regulatory.db.p7s"
S = "${WORKDIR}"
do_install() {
install -d ${D}${nonarch_base_libdir}/firmware
install -m 0644 ${WORKDIR}/regulatory.db ${D}${nonarch_base_libdir}/firmware/
install -m 0644 ${WORKDIR}/regulatory.db.p7s ${D}${nonarch_base_libdir}/firmware/
                        }

# Explicitly declare all files to be packaged

FILES_${PN}="${nonarch_base_libdir}/firmware/regulatory.db \
${nonarch_base_libdir}/firmware/regulatory.db.p7s"
```

- Save and exit the file.

2.2) Adding Firmware Files

- Create a new directory under the following path:
    - “**project-spec/meta-user/recipes-bsp/firmware/files/”**
- Place the firmware files into the file’s directory.
- Create the *BitBake* recipe file in firmware directory:
    - **vi <file_name.bb>**
    - **Example:** *vi firmware.bb*
- Now copy the below code. This will create the firmware directory under */lib* and copy the firmware files in this path *“lib/firmware/nxp”.*

```
DESCRIPTION = "Marvell SD9177 Wifi Firmware"
LICENSE = "CLOSED"
PR = "r0"

SRC_URI += "file://sduart_nw61x_v1.bin.se"

S = "${WORKDIR}"

do_install() {
install -d ${D}${nonarch_base_libdir}/firmware/nxp
   install -m 0644 ${WORKDIR}/sduart_nw61x_v1.bin.se
${D}${nonarch_base_libdir}/firmware/nxp/
}

# Explicitly declare all files to be packaged

FILES_${PN} = "${nonarch_base_libdir}/firmware/nxp/sduart_nw61x_v1.bin.se"
```

- Save and exit the file

2.3) Enabling Modules in RFS

- Open the *user-rootfsconfig* file located in: “**project-spec/meta-user/conf**”
- Add the following lines to include the regulatory and firmware recipes.
    - *CONFIG_< regulatory recipe name >*
    - *CONFIG_< firmware recipe name >*
    - **Example:** *CONFIG_regulatory-db*
      CONFIG_mrvl-firmware

- Run the following command to configure the root filesystem:
    - **petalinux-config -c rootfs**
- In the configuration menu, enable the desired recipes *(e.g., `regulatory-db and mrvl-firmware`)* under **User Packages**.
- To enable *`wpa_supplicant`*, navigate to: **Filesystem Packages → Networking → wpa-supplicant.**
- Save and exit the configuration.
- Run petalinux-config and navigate to: **Subsystem AUTO Hardware Settings → Serial Settings**
- Change the ***Serial stdin/stdout*** from *ps7_uart_1* to *ps7_uart_0*
- Save and exit the configuration.

2.4) Enabling Modules in Kernel

Ru**n "Petalinux-config -c kernel"** and enable *cfg80211* and *mac80211*.

2.5) Build the Project

- After completing all the above steps, build the *PetaLinux* project using:
    - **petalinux-build**
- After a successful build, you will find the output binary files such as image.ub, u-boot.elf, and others under the following directory: "**images/linux/"**

3) JTAG Boot Procedure to Flash Binaries on the Board

Below steps to boot the board using JTAG and flash the required binaries.

- Power on the board in JTAG mode. To enable JTAG mode, short the *MIO 5* pin to ground on the board. This action switches the board into JTAG mode.
- Next, open the **XSCT (Xilinx Software Command-line Tool)** console using either *PetaLinux* or *Vitis*. Once the XSCT console is open and connected to the board, check the available JTAG targets by entering the following command:
    - "**target"**
- This command will list the targets connected to the JTAG interface.
- Select target 2 by running the following command:
    - **ta 2**

- After selecting the target, load the FPGA bitstream file onto the board using the command:
    - **fpga <bit_file>**
- Replace <bit_file> with the actual bitstream file name.
- Then, load the First Stage Boot Loader (FSBL) by using the dow command as follows:
    - **dow <fsbl_file>**
    - For example**: "dow zynq_fsbl.elf"**
- After loading the FSBL, enter the command "**con"** to connect, and then enter "**stop"** to halt the processor.
- Next, load the Second Stage Boot Loader, which is the U-Boot file, with the following command:
    - **dow <u_boot_file>**
    - For example: "**dow u-boot.elf**"
- Once done, enter "**con"** to continue execution and observe the logs on the terminal. When ready, enter the "**stop**" command again to halt the processor.
- Now, load the main image file to a specific memory address using the dow command with the -data option:
    - **dow -data <image_file> 0x30000000**
    - For example: "**dow -data image.ub 0x30000000**"
- After the image is loaded, enter "**con"** to continue.
- If you access the prompt, press Ctrl + C repeatedly until the zynq prompt appears on the console.
- At the U-Boot prompt, use the following command to boot the image using the load addresses:
    - **bootm 0x30000000**
- This command boots the board using the image loaded at address 0x30000000.
- Finally, the board will boot up to the login prompt. Enter the username and password both as: root
- You are now logged in and ready to use the board.

4) QSPI Boot Procedure to Flash Binaries on the Board

The steps to flash and boot a Zynq board using QSPI via JTAG.

- Power on the board in JTAG mode. To enable JTAG mode, short the MIO pin 5 to ground on the board. This action switches the board into JTAG mode.
    - “**target”**
- This command will list the targets connected to the JTAG interface.
- Select target 2 by running the following command:
    - **ta 2**
- After selecting the target, load the FPGA bitstream file onto the board using the command:
    - **fpga <bit_file>**
- Replace <bit_file> with the actual bitstream file name.
- Then, load the First Stage Boot Loader (FSBL) by using the dow command as follows:
    - **dow <fsbl_file>**
    - For example**: “dow zynq_fsbl.elf”**
- After loading the FSBL, enter the command con to connect, and then enter stop to halt the processor.
- Next, load the Second Stage Boot Loader, which is the U-Boot file, with the following command:
    - **dow <u_boot_file>**
    - For example: “**dow u-boot.elf**”
- Meanwhile, go to your PetaLinux project and generate the BOOT.bin file using the petalinux-package command.
- After creating BOOT.bin, load it to a specific memory address using:
    - **dow -data BOOT.bin 0x08000000**
- Once the BOOT.bin is loaded, resume execution by entering:
    - **con**
- Then, quickly press **Enter** on the serial terminal to interrupt the U-Boot countdown and stop at the U-Boot prompt.
- At the U-Boot prompt, initialize the QSPI flash by running:
    - **sf probe 0 1000000 0**
- This command selects the QSPI flash and sets the clock to 1 MHz.
- Next, erase the QSPI flash memory (128 MB) using:
    - **sf erase 0 0x08000000**
- Then, write the BOOT.bin to flash from RAM with the following command:
    - **sf write 0x08000000 0 0x08000000**

- After the write completes, power off the board. Then, power it back on in QSPI boot mode. The board should now boot in QSPI boot mode.
- If you access the prompt, press Ctrl + C repeatedly until the zynq prompt appears on the console.
- At the U-Boot prompt, use the following command to boot the image using the load addresses:
    - **bootm <Image_load_addresses>**
    - For Example**: bootm 0x10000000**
- This command boots the board using the image loaded at address 0x30000000.
- Finally, the board will boot up to the login prompt. Enter the username and password both as: root
- You are now logged in and ready to use the board.

5) Manual Loading of Wi-Fi Kernel Modules

- After booted the bard, Verify that the following kernel module files are present in the directory: /lib/modules/5.4.0-xilinx-v2020.2/extra
    - mlan.ko
    - moal.ko
- Load the modules manually using the insmod command
    - **insmod maln.ko**
    - **insmod moal.ko**
- After loading moal.ko, kernel logs will appear in the console indicating the driver has been successfully loaded.
- These logs include messages such as:

  wlan: Loading MWLAN driver

  vendor=0x0471 device=0x0205 class=0 function=1

  Attach moal handle ops, card interface type: 0x109

  Request firmware: nxp/sduart_nw61x_v1.bin.se

  WLAN FW is active

Register NXP 802.11 Adapter mlan0

wlan: version = SDIW612---18.99.3.p23.6-MM6X18505.p14-GPL-(FP92)

wlan: Driver loaded successfully

- These messages confirm that the firmware has been downloaded, devices are registered (e.g., mlan0, uap0, wfd0), and the driver is ready for use.

6) Wi-Fi Connection Process on the Board

When the board boots, you should see Wi-Fi-related logs in the console output. These logs typically include the Wi-Fi device ID and confirmation that the firmware has been successfully loaded.

- After the board has fully booted, open a terminal and run the following command to list all available network interfaces:
  - **ifconfig -a**
- Look for the WiFi interface, usually named mlan0. To bring this interface up, run:
  - **ifconfig mlan0 up**
- To stop unwanted kernel messages during scan, run:
  - **echo "0 0 0 0" > /proc/sys/kernel/printk**
- Next, Start the wpa_supplicant service in the background to enable WiFi management via wpa_cli:
  - **wpa_supplicant -B -i mlan0 -c /etc/wpa_supplicant.conf**
- To scan for available WiFi networks, run:
  - **wpa_cli -i mlan0 scan**
- After the scan completes, display the results with:
  - **wpa_cli -i mlan0 scan_results**
- This command will list nearby WiFi networks. If your target network does not appear, scan again and review the results.
- To initiate a connection, add a new network configuration:
  - **wpa_cli -i mlan0 add_network**
- This will return a network ID (e.g., 0). Use this ID in the following steps to set the SSID and password.
- Set the SSID of your WiFi network:
  - **wpa_cli -i mlan0 set_network <id> ssid '"<your_wifi_name>"'**
- Set the password for the WiFi network:

- **wpa_cli -i mlan0 set_network <id> psk '"<your_password>"'**

- Enable the network using:
  - **wpa_cli -i mlan0 enable_network <id>**
- To save the configuration :
  - **wpa_cli save_config**
- Check the connection status:
  - **wpa_cli status**
- If the status does not show "COMPLETED", try listing the configured networks:
  - **wpa_cli list_networks**
- Select the desired network manually:
  - **wpa_cli select_network <id>**
- Save the configuration again to ensure settings are stored:
  - **wpa_cli save_config**
- Run the status command again to confirm the connection. If the board is still not connected, verify that your WiFi router or hotspot is active and within range.
- To obtain an IP address dynamically, use:
  - **udhcpc -i mlan0**
- Alternatively, you can assign a static IP if needed.
- Finally, test the connection by pinging an external server such as Google DNS:
  - **ping 8.8.8.8**
- Or ping another device on the same local network to confirm connectivity.

7) Sample Logs

Below are the tested logs captured from the board. They show clear **inputs** and **outputs** during the process of enabling WiFi, connecting to a network, setting IP configuration, and accessing a remote host via SSH.

root@wifi_qspi:~# ifconfig mlan0 up

root@wifi_qspi:~# wpa_cli -i mlan0 scan_results

bssid / frequency / signal level / flags / ssid
2a:6b:35:63:29:d7 5765 -62 [WPA2-PSK-CCMP][WPS][ESS] LAPTOP
40:ae:30:d7:06:66 5765 -64 [WPA2--CCMP][ESS] Vconnet-TPLink-5G
42:ae:30:67:06:64 5765 -64 [WPA2-PSK-CCMP][ESS] Guest_Wifi-5G

c8:a6:08:95:d5:d0 5600 -63 [WPA2-PSK-CCMP][ESS] Vconnectech
42:ae:30:77:06:64 5765 -63 [WPA2-PSK-CCMP][ESS]
c8:a6:08:95:ca:60 5680 -67 [WPA2-PSK-CCMP][ESS] Vconnectech
40:ae:30:d7:06:64 2447 -65 [WPA2--CCMP][ESS] Vconnet-TPLink
42:ae:30:17:06:64 2447 -65 [WPA2-PSK-CCMP][ESS] Guest_Wifi
c8:a6:08:55:d5:d0 2412 -70 [WPA2-PSK-CCMP][ESS] Vconnectech
c8:a6:08:55:ca:60 2462 -76 [WPA2-PSK-CCMP][ESS] Vconnectech
7e:4d:8f:7e:a9:03 2412 -83 [WPA2-PSK-CCMP][WPS][ESS][P2P]
DIRECT-03-HP Laser 150nw
42:ae:30:27:06:64 2447 -65 [WPA2-PSK-CCMP][ESS]
98:ba:5f:36:b3:de 2422 -98 [WPA2-PSK-CCMP][WPS][ESS] Anthe Restaurant
3c:64:cf:ce:57:ca 2462 -96 [WPA-PSK-CCMP][WPA2-PSK-CCMP][ESS]
RESIGN 2.0
00:31:92:1b:24:c2 2412 -98
[WPA-PSK-CCMP+TKIP][WPA2-PSK-CCMP+TKIP][ESS] TRS

root@wifi_qspi: ~# wpa_cli add_network
Selected interface 'mlan0'
1
root@wifi_qspi: ~# wpa_cli set_network 1 ssid '"Vconnectech"'
Selected interface 'mlan0'
OK
root@wifi_qspi: ~# wpa_cli set_network 1 psk '"Vconnectech@123"'
Selected interface 'mlan0'
OK
root@wifi_qspi:~# wpa_cli enable_network 0
Selected interface 'mlan0'
OK
root@wifi_qspi:~# wpa_cli enable_network 1
Selected interface 'mlan0'
OK
root@wifi_qspi:~# wpa_cli status
Selected interface 'mlan0'
bssid=c8:a6:08:95:ca:60
freq=5680
ssid=Vconnectech
id=1

```
mode=station
wifi_generation=6
pairwise_cipher=CCMP
group_cipher=CCMP
key_mgmt=WPA2-PSK
wpa_state=COMPLETED
address=10:32:2c:70:99:4a
uuid=033ad294-c8f1-5100-b34e-749b1c075070
ieee80211ac=1
root@wifi_qspi:~# ifconfig mlan0 192.168.10.12 netmask 255.255.255.0 up
root@wifi_qspi:~# route add default gw 192.168.10.1
root@wifi_qspi:~# ifconfig mlan0
mlan0     Link encap:Ethernet  HWaddr 10:32:2C:70:99:4A
          inet addr:192.168.10.12  Bcast:192.168.10.255  Mask:255.255.255.0
          inet6 addr: fd14:5d30:5573:7fdd:1232:2cff:fe70:994a/64 Scope:Global
          inet6 addr: fe80::1232:2cff:fe70:994a/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:67 errors:0 dropped:1 overruns:0 frame:0
          TX packets:33 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4362 (4.2 KiB)  TX bytes:6276 (6.1 KiB)

root@wifi_qspi:~# ping 192.168.10.81
PING 192.168.10.81 (192.168.10.81): 56 data bytes
64 bytes from 192.168.10.81: seq=0 ttl=64 time=4.392 ms
64 bytes from 192.168.10.81: seq=1 ttl=64 time=3.403 ms
64 bytes from 192.168.10.81: seq=2 ttl=64 time=3.471 ms
^C
--- 192.168.10.81 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 3.403/3.755/4.392 ms
root@wifi_qspi:~# ssh admin1@192.168.10.81

Host '192.168.10.81' is not in the trusted hosts file.
(ecdsa-sha2-nistp256 fingerprint sha1!!
91:7f:b9:15:50:19:be:dc:ef:de:c3:b3:d0:bd:ff:24:d6:56:10:ae)
Do you want to continue connecting? (y/n) y
```

admin1@192.168.10.81's password:
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.5.0-45-generic x86_64)

** Documentation: https://help.ubuntu.com*
** Management: https://landscape.canonical.com*
** Support: https://ubuntu.com/pro*

Expanded Security Maintenance for Applications is not enabled.

127 updates can be applied immediately.
101 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

34 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

New release '24.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Tue May 20 15:02:03 2025 from 192.168.10.126
admin1@VC119-pk:~$
admin1@VC119-pk:~$