Getting started with AIROC™ Wi-Fi & Bluetooth® combo chip on iMX8 Nano Developer's Kit V3 in Linux

About this document

Scope and purpose

This guide provides step-by-step instructions to configure the Infineon's AIROC™ Wi-Fi & Bluetooth® combo chip from a host, load the WLAN driver, and establish a Wi-Fi connection between an Access Point (AP)/SoftAP and STA. Further sections describe flashing a new image and setting up the toolchain for image customization for advanced users. This guide also includes step-by-step instructions to configure and validate Bluetooth® and Bluetooth® Low Energy functionality of the Infineon's AIROC™ Wi-Fi & Bluetooth® combo chip.

Intended audience

This document is intended for customers and FAEs using the AIROC™ Wi-Fi & Bluetooth® combo chip on iMX8 Nano Developer's Kit V3 as host.

Table of contents

Table of contents

Introduction

1 Introduction

This guide provides step-by-step instructions to configure and load the WLAN driver, and establish a Wi-Fi connection between a SoftAP/AP and STA. This guide also includes getting started information for Bluetooth® and Bluetooth® Low Energy (LE) functionality. The necessary WLAN-related files are provided as an attachment along with this guide. Contact your local Infineon FAE or sales representative to get the files.

Depending on your context, navigate to the section most relevant to you as follows:

If you…	Do this
Have a host preloaded with Linux 6.1.36 or 5.15.32	See Overview of software components
Want to flash Linux 6.1.36 or 5.15.32	See Flashing the prebuild package
Want to build a custom Linux image	See Custom image generation and toolchain setup.

Section 2 Overview of hardware components gives a glimpse of all the hardware requirements to quickly set to the CYW55573 and CYW43022 combo chip working along with the interface and connecting it to the iMX8 Nano platform.

Section 3 Overview of software components lists and describes the files available in the package that are required to load the firmware and driver along with the steps and examples to follow to load the secure Infineon image into the Infineon (AIROC™ CYW55573 and AIROC™ CYW43022) Wi-Fi & Bluetooth® combo chip.

Section 4 Configure iMX8 Nano host to download mode describes the hardware changes required to flash a new image in the host board.

Section 5 Custom image generation and toolchain setup describes how to set up the toolchain for customizing the image if you want to develop your custom image and set up a toolchain for tuning the image.

Section 6 Flashing the prebuild package describes the image, firmware, and driver package for the module and steps to flash the image onto the host platform (iMX8 Nano kit).

Section 7 Building the FMAC driver from backports (Kernel version lower than 6.1.145) describes the procedure to compile the FMAC drivers from the backporting process.

Section 8 Wi-Fi setup describes how to make the Infineon (AIROC™ CYW55573 and AIROC™ CYW43022) AIROC™ CYW55573 Wi-Fi & Bluetooth® combo chip as Station (STA) and connect to AP with security like open, wpa2-psk, and wpa3-psk. Similarly, it explains how to make the Infineon (AIROC™ CYW55573 and AIROC™ CYW43022) AIROC™ CYW55573 Wi-Fi & Bluetooth® combo chip as SoftAP. Additionally, it describes how to measure the throughput. The iPerf tool is used to measure the throughput data like bandwidth, loss, and other parameters.

Section 9 Bluetooth® provides a brief about Bluetooth® development environment setup. This section gives steps to create the software setup and hardware setup for Bluetooth® development. This section explains about the release package and application software support for Bluetooth® development. This section also explains about the Manufacturing Bluetooth® Tool and BlueZ setup in our Bluetooth® development environment setup.

Overview of hardware components

2 Overview of hardware components

2.1 iMX8M Nano Developer's Kit V3

- PC with Windows or Linux operating system
- Host platform – iMX8M Nano Developer's Kit V3 from Embedded Artist
- 12-V power supply
- AIROC™ Wi-Fi & Bluetooth® combo chip
- Micro-B to USB-A cable
- Ethernet LAN cable

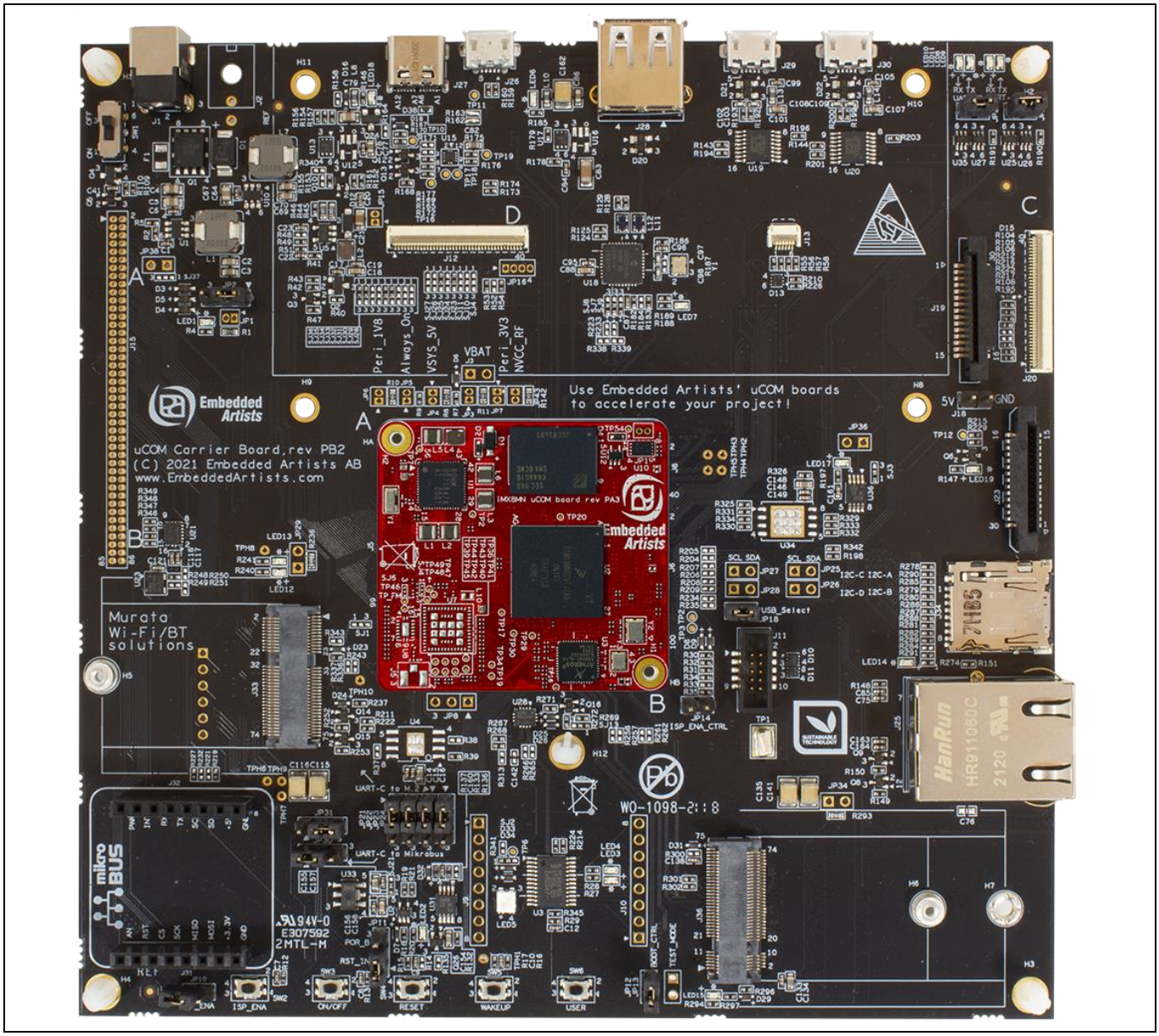

Figure 1 iMX8M Nano Developer's Kit V3 board

Overview of hardware components

Figure 2 AIROC™ CYW55573 Wi-Fi & Bluetooth® combo chip

Figure 3 AIROC™ CYW43022 Wi-Fi & Bluetooth® combo chip

Figure 4 AIROC™ CYW55513 Wi-Fi & Bluetooth® combo chip

Overview of hardware components

2.2 Host overview

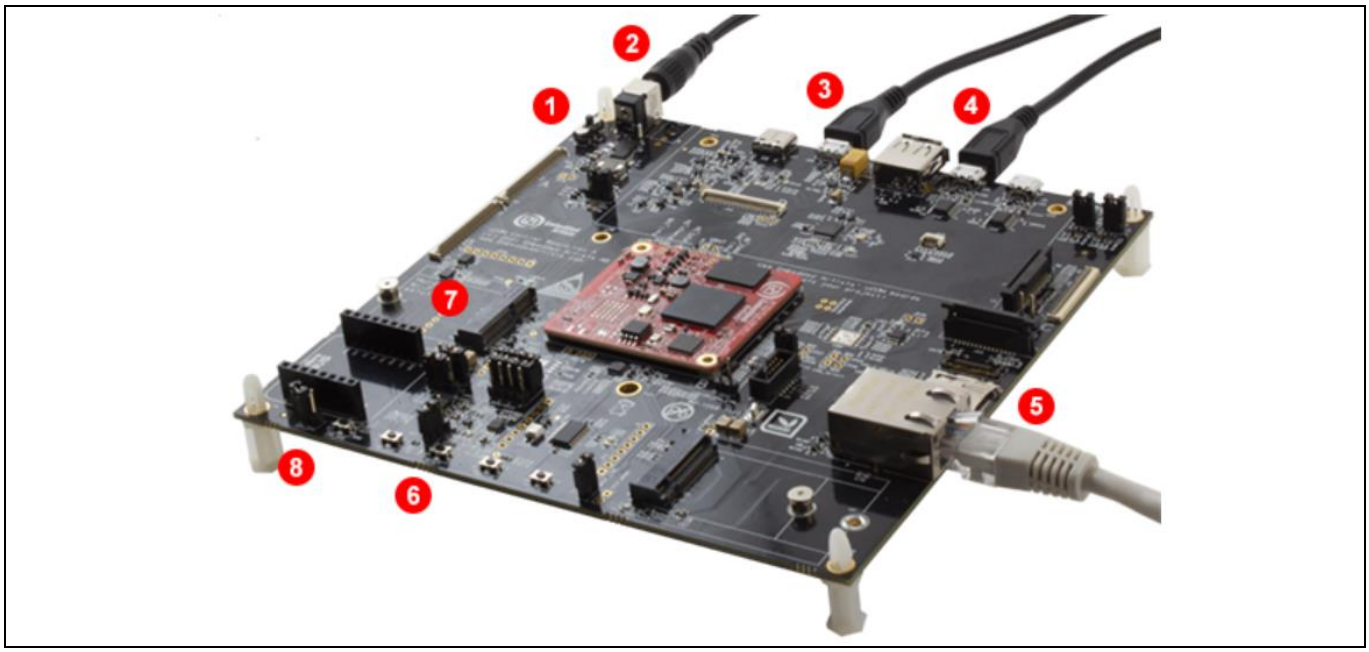

Figure 5 µCOM Developer's Kit board overview

1. Power ON/OFF switch
2. Power supply
3. Micro-B to A USB cable connected to USB OTG
4. Micro-B to A USB cable for the console
5. Network cable connected to RJ45 connector
6. Reset button
7. M.2 connector for Wi-Fi/Bluetooth®
8. ISP enable jumper (OTG boot mode)

2.3 Hardware setup

Connect the AIROC™ Wi-Fi & Bluetooth® combo chip to the M.2 E-key connector (J33) on the iMX8M Nano Developer's Kit V3 board. The M.2 E-key interface has an SDIO that implements a Wi-Fi/Bluetooth® interface.

Overview of hardware components

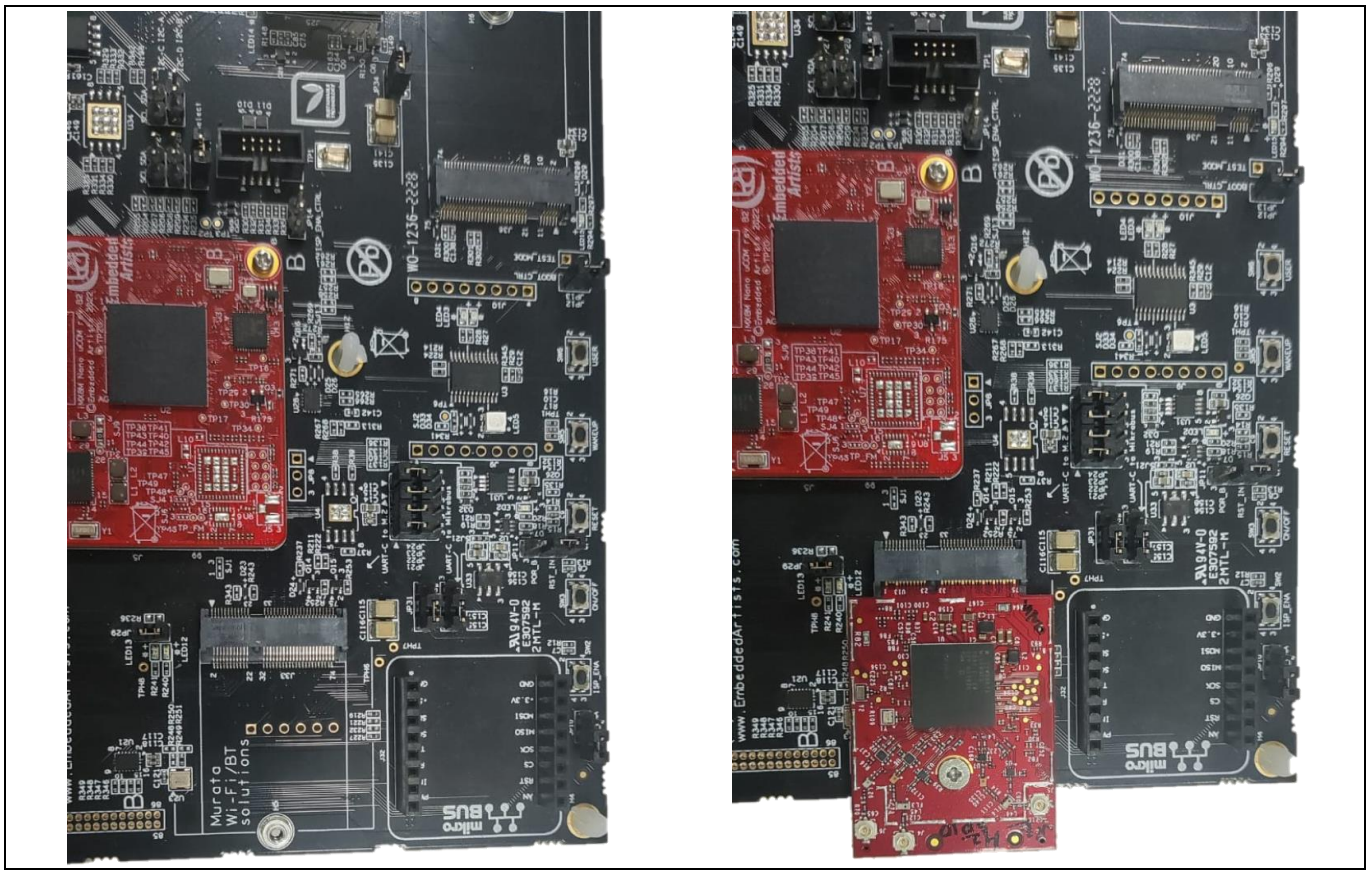

Figure 6 AIROC™ Wi-Fi & Bluetooth® combo chip connected to iMX8M Nano Developer's Kit V3 board

3 Overview of software components

The following files are required to load the firmware and driver.

Table 1 Files required to load firmware and driver

File name	File type	Description
compat.ko *cfg80211.ko* *brcmutil.ko* *brcmfmac.ko* *brcmfmac-cyw.ko*	Driver file	The loadable kernel modules file (*.ko* – Kernel Object) is used by the Linux kernel. This contains a program code that extends the functionality of the Linux kernel, such as code for a computer device driver to load a kernel module, using the `insmod` utility. This utility receives as a parameter and the path to the **.ko* file, unloading the module from the kernel is done using the `rmmod` command, which receives the module name as a parameter.
cyfmac55573-sdio.clm_blob *cyfmac43022-sdio.clm_blob* *cyfmac55500-sdio.clm_blob*	Clm_blob file	The firmware for Infineon Wi-Fi chipsets includes information on regulatory constraints. For certain modules, this information is kept separately in a binary form known as "CLM blob". Country Locale Matrix (CLM) is the Infineon-maintained database that is the regulatory configuration (target power outputs) of Wi-Fi modules concerning countries, bands, data rates, and channels. The CLM Blob file is loaded by the FMAC driver at boot time.
cyfmac55573-sdio.trxse *cyfmac43022-sdio.trxs* *cyfmac55500-sdio.trxse*	Secure firmware file	Secure and encrypted Wi-Fi firmware (TRXSE). This is the secure firmware with security features enabled.
cyfmac55573-sdio.txt *cyfmac43022-sdio.txt* *cyfmac55500-sdio.txt*	NVRAM file	Some devices store the NVRAM, which is needed in addition to the firmware by the driver. *Note: The NVRAM file available in this package is coupled with the latest firmware available in the release. It is not backward compatible with the older release.*

4 Configure iMX8 Nano host to download mode

To download and deploy a new image, update the kernel or update the U-Boot, and set the iMX8M Nano Developer's Kit V3 board in "Image Download Mode" by following these steps:

1. Enable ISP-Enabled Mode by shorting the ISP_ENA jumper as shown in Figure 7.
2. Connect a USB cable between the connector (USB for Image Download) and your PC.

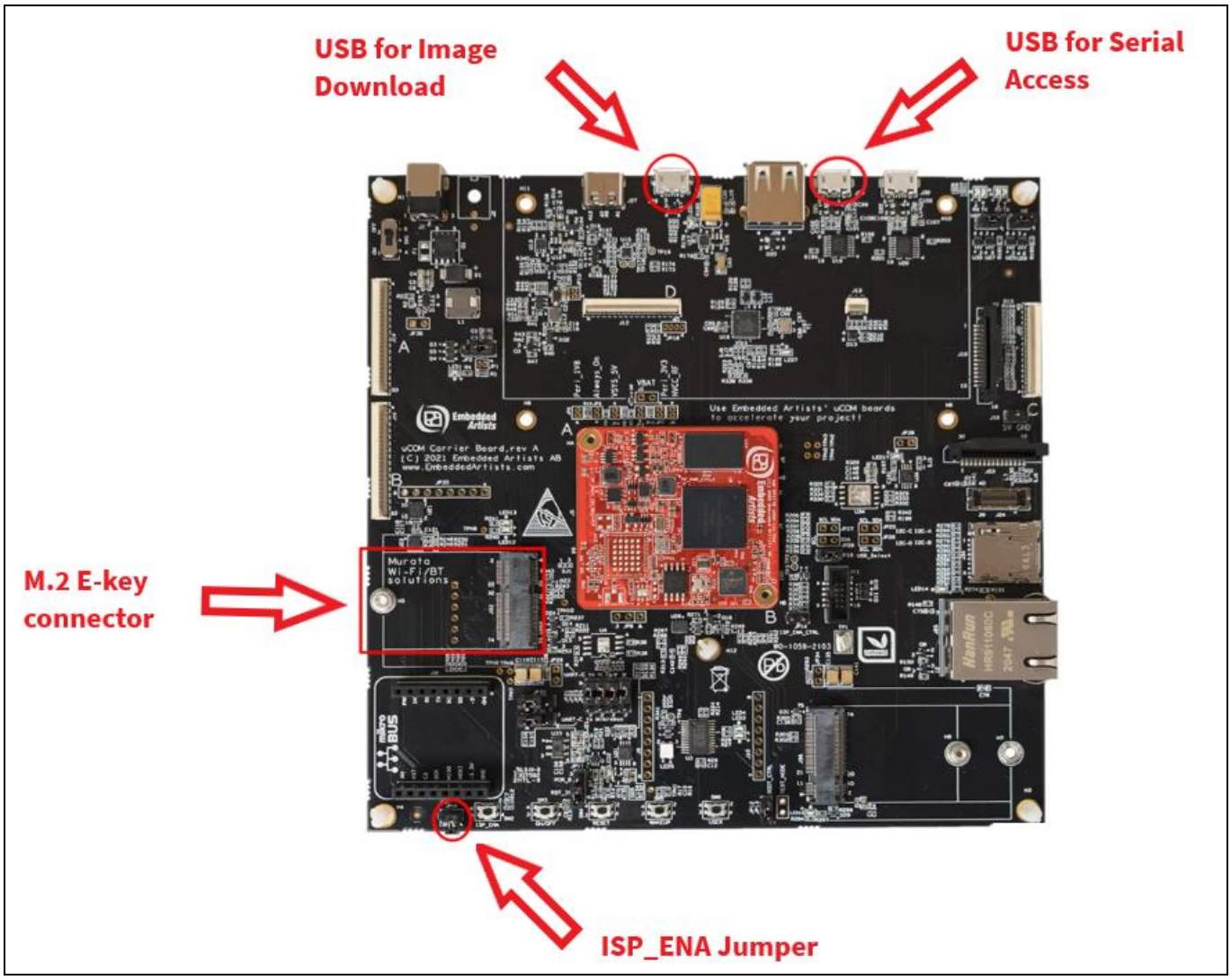

Figure 7 iMX8 connections and ports

5 Custom image generation and toolchain setup

If the toolchain setup is not required, please proceed to Building the kernel zImage.

5.1 Linux host setup

The Yocto build system requires a Linux host machine. The minimum available hard disk space is 50 GB; however, it is recommended that the host machine has at least 120 GB to be able to build the largest image/distribution.

The instructions in this document are tested on Ubuntu 18.04.

5.2 Required packages

The Yocto Project requires several packages to be installed on the host machine.

Yocto dependency installation

```
sudo apt-get update
sudo apt-get install openssh-server

sudo service ssh restart


sudo apt-get update


sudo apt-get install gawk wget git-core diffstat unzip texinfo \
gcc-multilib build-essential chrpath socat \
libsdl1.2-dev xterm sed cvs subversion \
coreutils texi2html docbook-utils python-pysqlite2 help2man make \
gcc g++ desktop-file-utils libgl1-mesa-dev libglu1-mesa-dev \
mercurial autoconf automake groff curl lzop asciidoc u-boot-tools
```

1. Create a directory for the tool. The following command creates a directory named *bin* in your home folder.

```
mkdir ~/bin
```

2. Download the tool using the following command:

```
curl http://commondatastorage.googleapis.com/git-repodownloads/repo > ~/bin/repo
```

3. Make the tool executable:

```
chmod a+x ~/bin/repo
```

4. Add the directory to the PATH variable. Add the following line to your *.bashrc* file so the path is available in each started shell/terminal:

```
echo "export PATH=~/bin:$PATH" >> ~/.bashrc
source ~/.bashrc
```

Custom image generation and toolchain setup

5.3 Download Yocto recipes

The Yocto Project consists of many recipes used when building an image. These recipes come from several repositories; the repo tool is used to download these repositories. A branch must be selected of the *ea-yocto-base* repository. This table lists the available branches.

Table 2 Branches of ea-yocto-base repo

Branch name	Description
ea-6.12.3	u-boot: 2024.04. Linux: 6.12.3
ea-6.6.23	u-boot: 2024.04. Linux: 6.6.23
ea-6.1.36	u-boot: 2022.04. Linux: 6.1.36
ea-5.15.32	u-boot: 2022.04. Linux: 5.15.32
ea-5.10.72	u-boot: 2021.04. Linux: 5.10.72
ea-5.10.35	u-boot: 2021.04. Linux: 5.10.35
ea-5.4.47	u-boot: 2020.04. Linux: 5.4.47
ea-5.4.24	u-boot: 2020.04. Linux: 5.4.24
ea-4.14.98	u-boot: 2018.03. Linux: 4.14.98
ea-4.14.78	u-boot: 2018.03. Linux: 4.14.78
ea-4.9.123	u-boot: 2017.03. Linux: 4.9.123
ea-4.9.11_1.0.0	u-boot: 2016.03. Linux: 4.9.11
ea-4.1.15_2.0.0	u-boot: 2016.03. Linux: 4.1.15

1. Create a directory for the downloaded files (*ea-bsp* in the following example):

```
mkdir ea-bsp
cd ea-bsp
```

2. Configure the Git if you have not already done so. Change "Your name" to your actual name and "Your e-mail" to your e-mail address.

```
git config --global user.name "Your name"
git config --global user.email "Your e-mail"
```

3. Initialize the repo. The file containing all the required repositories is downloaded in this step. Change <*selected branch*> to a branch name according to Table 2.
   The following example uses the *ea-6.12.3 branch:*

```
repo init -u https://github.com/embeddedartists/ea-yocto-base -b ea-6.12.3
```

   The following example uses the *ea-6.6.23 branch:*

```
repo init -u https://github.com/embeddedartists/ea-yocto-base -b ea-6.6.23
```

   The following example uses the *ea-6.1.36 branch:*

```
repo init -u https://github.com/embeddedartists/ea-yocto-base -b ea-6.1.36
```

4. Start to download the files:

```
repo sync
```

All files have now been downloaded into the *ea-bsp* directory. Most of the files will be available in the subdirectory called *sources*.

5.4 Initialize the build

1. Enter the following command to change your directory automatically to *build-imx8mqea-com-wayland.*

```
DISTRO=fsl-imx-wayland MACHINE=imx8mnea-ucom source ea-setup-release.sh -b build-imx8mnea-ucom-wayland
```

2. Edit *conf/local.conf* to enable SSH server:

```
- EXTRA_IMAGE_FEATURES = "debug-tweaks"
+ EXTRA_IMAGE_FEATURES = "debug-tweaks ssh-server-openssh"
```

5.5 Custom iMX toolchain

1. Build an image to create the toolchain:

```
bitbake meta-toolchain
```

2. This creates a file *build-imx8mnea-ucom-wayland/tmp/deploy/sdk*. Go to this location to install the toolchain:

   For example, for kernel version 6.12.3:

```
cd build-imx8mnea-ucom-wayland/tmp/deploy/sdk
sudo ./fsl-imx-wayland-glibc-x86_64-meta-toolchain-armv8a-imx8mnea-ucom-toolchain-6.12-walnascar.sh
```

   For kernel version 6.6.23:

```
sudo ./fsl-imx-wayland-glibc-x86_64-meta-toolchain-armv8a-imx8mnea-ucom-toolchain-6.6-scarthgap.sh
```

   For kernel version 6.1.36:

```
sudo ./fsl-imx-wayland-glibc-x86_64-meta-toolchain-armv8a-imx8mnea-ucom-toolchain-6.1-mickledore.sh
```

   This will result in the toolchain being installed in */opt/fsl-imx-wayland/<version>*

3. Source the environment variables:

   For kernel version 6.12.3:

```
source /opt/fsl-imx-wayland/6.12-walnascar/environment-setup-armv8a-poky-linux
```

   For kernel version 6.6.23:

```
source /opt/fsl-imx-wayland/6.6-scarthgap/environment-setup-armv8a-poky-linux
```

   For kernel version 6.1.36:

Custom image generation and toolchain setup

```
source /opt/fsl-imx-wayland/6.1-mickledore/environment-setup-armv8a-poky-linux
```

5.6 Building the image

Now that everything is set up, the build can begin. The construction of the image *ea-image-base* is demonstrated in the following example. Note that creating an image may take many hours, depending on the host computer's capability.

```
bitbake ea-image-base
```

The output directory of the build will appear in *~/ea-bsp/build-imx8mnea-ucom xwayland/tmp/deploy/images/imx8mnea-ucom.*

5.7 Building the kernel zImage

1. For the 6.12.3 kernel, download the kernel source code from iMX:

```
$ git clone https://github.com/embeddedartists/linux-imx.git
$ cd linux-imx
$ git checkout bbcfde34f99f18d5ad49a6f87b05f3263666d12c
```

   For the 6.6.23 kernel, use `git checkout ea_6.6.y`.

   For the 6.1.36 kernel, use `git checkout ea_6.1.y`.

2. Set up the build environment:

```
$ source /opt/fsl-imx-wayland/6.12-walnascar/environment-setup-armv8a-poky-linux
```

3. Run the following command to avoid linker errors:

```
$ unset LDFLAGS
```

4. Edit the default configuration to disable bcmdhd and build cfg80211 as a module:

```
$ vi arch/arm64/configs/ea_imx8_defconfig
CONFIG_CFG80211=m
CONFIG_BCMDHD=n
CONFIG_USB_SERIAL=y
CONFIG_USB_SERIAL_CONSOLE=y
CONFIG_USB_SERIAL_GENERIC=y
CONFIG_USB_SERIAL_FTDI_SIO=y
CONFIG_ASYMMETRIC_KEY_TYPE=y
CONFIG_ASYMMETRIC_PUBLIC_KEY_SUBTYPE=y
CONFIG_X509_CERTIFICATE_PARSER=y
CONFIG_PKCS7_MESSAGE_PARSER=y

#3.1 Disable cfg80211 regdb for the kernel above v5.4.18

CONFIG_CFG80211_REQUIRE_SIGNED_REGDB=n
CONFIG_CFG80211_USE_KERNEL_REGDB_KEYS=n
```

5. For better low-power performance, make sure to apply the kernel patches for iMX8 from the community, which contains the custom mmc patches.
6. For **kernel version 6.6.23 or 6.12.3**, apply the respective Infineon FMAC patches for iMX8 from the community.
7. Build the kernel.

Custom image generation and toolchain setup

```
$ make clean
$ make ea_imx8_defconfig
$ make
```

The built kernel is available in *arch/arm64/boot/Image.*

For Kernel 6.6 only, use the `make modules` command to generate the kernel modules. Then run the following command to install the generated modules into a build machine.

```
make modules_install INSTALL_MOD_PATH=<path the modules built would be available>
```

Replace the generated file from the build machine to the */lib/modules* folder of imx8 host.

6 Flashing the prebuild package

1. Download the appropriate iMX8 Nano image from here to your PC.
2. For example, for kernel version 6.6.23, unzip the *uuu_imx8mn_ucom_6.6.23.zip* image package.
3. Copy the Image file and *imx8mn-ea-ucom-kit_v3.dtb* generated by following the steps in Custom image generation and toolchain setup to *uuu_imx8mn_ucom_6.6.23\files*.
4. Edit the *kernel.uuu* file in the *uuu_imx8mn_ucom_6.6.23\ folder* with the image filename generated by following the steps in Custom image generation and toolchain setup.
5. Put the iMX8 board in Image Download Mode.
6. Power ON the iMX8M Nano Developer's Kit.
7. Use the Universal Update Utility (UUU) tool to download the images to the host board.
8. After connecting the hardware to your PC, open the command prompt or terminal and navigate to *uuu_imx8mn_ucom_5.15.32* for the 5.15.32 kernel.
9. Use the following command to check whether the board is recognized by the PC.

```
uuu -lsusb
```

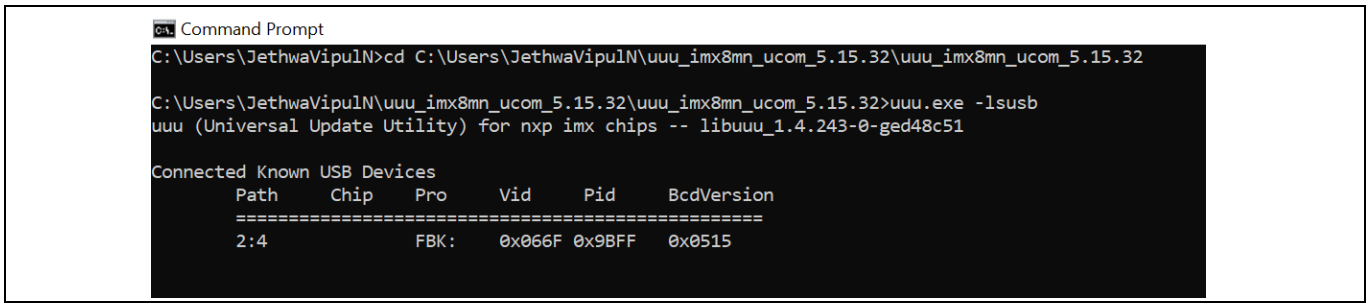

Figure 8 Command to check if the board is recognizable

Note: If the board already has the Linux 5.15.32 kernel, you can skip to Step 13.

10. Download the full image using the following command.

```
uuu.exe full_wic.uuu
```

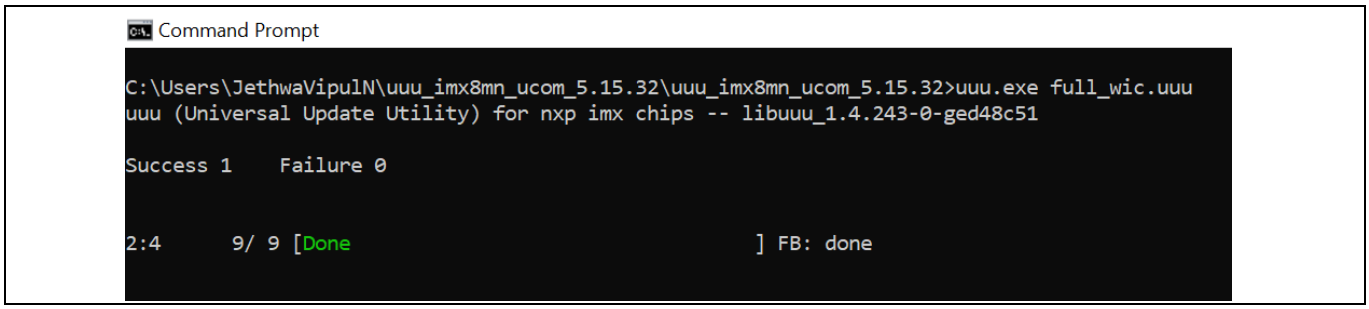

Figure 9 Programing full image using UUU Tool

Note: After running the uuu.exe command, power OFF and power ON the board before entering another uuu.exe command.

Flashing the prebuild package

11. Download the device tree using the following command:

```
uuu.exe kernel.uuu
```

Figure 10 Programing kernel using UUU tool

12. After the image is downloaded to the iMX8M Nano Developer's Kit, power OFF the device and disconnect the ISP_ENA jumper to bring the board to Normal Mode.
13. Connect to the serial port of the hardware and open the COM port in the terminal application at a baud rate of **115200**.
14. Power ON the board and press **Enter** or any key on the terminal to stop autoboot and enter U-BOOT mode.
15. Enter the following command and confirm that *fdt* is used:

```
u-boot=> printenv fdt_file
fdt_file=imx8mn-ea-ucom-kit_v3.dtb
```

16. Enter the following command to set the *.dtb* file you want (*imx8mn-ea-ucom-kit_v3.dtb -> SDR104/OOB interrupt, imx8mn-ea-ucom-kit_v3_ifx.dtb ->inband interrupt/BT*):

```
u-boot=> setenv fdt_file imx8mn-ea-ucom-kit_v3.dtb
```

17. Enter the following command to save the changes:

```
u-boot=> saveenv
```

18. Enter the following command to confirm the changes:

```
u-boot=> printenv fdt_file
```

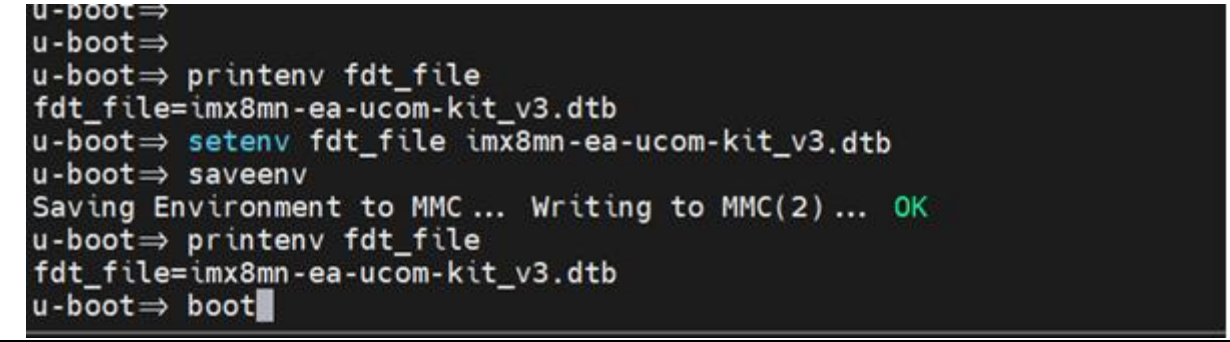

Figure 11 Changing DTB file for SDIO versions

19. Enter the following command to continue the booting process:

```
u-boot=> boot
```

Flashing the prebuild package

20. When the booting process is complete, you will be prompted to log in. Enter the following login credentials:
    - Username: **root**
    - Password: **pass**
21. After login, check whether the AIROC™ CYW55573 Wi-Fi & Bluetooth® combo chip is connected to the M.2 E-key connector on the iMX8M Nano Developer's Kit V3 board and is recognized by the OS using the following command. It will return the details of the chip with SDIO mode and the MultiMediaCard ID connected.

    For example, Figure 12 shows that the CYW55573 is connected to mmc0.

```
dmesg | grep SDIO
```

```
root@imx8mnea-ucom:~# dmesg | grep SDIO
[    2.590806] mmc0: new ultra high speed SDIO card at address 0001
root@imx8mnea-ucom:~#
```

Figure 12 Command to check if CYW55573 is connected to the host

22. Use the following command to confirm the SDIO settings.

```
root@imx8mnea-ucom:~# cat /sys/kernel/debug/mmc0/ios
```

```
root@imx8mnea-ucom:~# cat /sys/kernel/debug/mmc0/ios
clock:          200000000 Hz
actual clock:   200000000 Hz
vdd:            21 (3.3 ~ 3.4 V)
bus mode:       2 (push-pull)
chip select:    0 (don't care)
power mode:     2 (on)
bus width:      2 (4 bits)
timing spec:    6 (sd uhs SDR104)
signal voltage: 1 (1.80 V)
driver type:    0 (driver type B)
root@imx8mnea-ucom:~#
```

Figure 13 SDIO MMC details

7 Building the FMAC driver from backports (Kernel version lower than 6.1.145)

1. Get the FMAC package from the Infineon community. Contact your local Infineon FAE or sales representative to get the *CYW55513.zip* release package.

```
cd ea-bsp/kernel/linux-imx
$ copy the fmac zip file to the machine
$ unzip cypress-fmac-*.zip
$ tar zxvf cypress-backports-*.tar.gz
$ cd v5.15-backports
```

2. Open a new terminal but do not source the Yocto toolchain yet and run the following command:

```
$ MY_KERNEL=<kernel path>
# Example: MY_KERNEL= ea-bsp/kernel/linux-imx
$ make KLIB=$MY_KERNEL KLIB_BUILD=$MY_KERNEL defconfig-brcmfmac
```

3. Source the Yocto toolchain (for cross-compile) and build the FMAC driver:

```
$ source /opt/fsl-imx-xwayland/5.15-kirkstone/environment-setup-armv8a-poky-linux
$ unset LDFLAGS
$ make KLIB=$MY_KERNEL KLIB_BUILD=$MY_KERNEL modules
```

The output modules can be found in the following:

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

Building the FMAC driver from backports (Kernel version lower than 6.1.145)

7.1 Debugging the FMAC driver

1. Enable the debug feature in backports source code, and enable CPTCFG_BRCMDBG and CONFIG_DEBUG_FS in the *defconfig* file:

```
$ vi defconfigs/brcmfmac
```

2. Add the following lines to the file:

```
$ CPTCFG_BACKPORTED_DEBUG_INFO=y
$ CPTCFG_BRCM_TRACING=y
$ CPTCFG_BRCMDBG=y
```

3. Rebuild the FMAC driver per instructions in the Section 5.6 for kernel 6.1.36 or Section 7 for backports.
4. Enable the *brcmfmac* debug log:

```
$ echo 8 > /proc/sys/kernel/printk
$ insmod brcmfmac.ko debug=0x00100006      /*(to enable TRACE, INFO, and
WIFI_FW_LOG, for example)*/
```

A complete list of message levels is listed in *drivers/net/wireless/broadcom/brcm80211/brcmfmac/debug.h*.

```
/* message levels */
#define BRCMF_TRACE_VAL 0x00000002
#define BRCMF_INFO_VAL 0x00000004
#define BRCMF_DATA_VAL 0x00000008
#define BRCMF_CTL_VAL 0x00000010
#define BRCMF_TIMER_VAL 0x00000020
#define BRCMF_HDRS_VAL 0x00000040
#define BRCMF_BYTES_VAL 0x00000080
#define BRCMF_INTR_VAL 0x00000100
#define BRCMF_GLOM_VAL 0x00000200
#define BRCMF_EVENT_VAL 0x00000400
#define BRCMF_BTA_VAL 0x00000800
#define BRCMF_FIL_VAL 0x00001000
#define BRCMF_USB_VAL 0x00002000
#define BRCMF_SCAN_VAL 0x00004000
#define BRCMF_CONN_VAL 0x00008000
#define BRCMF_BCDC_VAL 0x00010000
#define BRCMF_SDIO_VAL 0x00020000
#define BRCMF_MSGBUF_VAL 0x00040000
#define BRCMF_PCIE_VAL 0x00080000
#define BRCMF_FWCON_VAL 0x00100000
```

5. Continuously print out debug messages:

```
dmesg -w
```

8 Wi-Fi setup

8.1 Load the drivers and firmware

Do the following to load the firmware:

1. Copy the kernel modules generated to the locally created folder *CYW5557x/CYW43022*.
   a) For 6.6.23/6.12.3 - from the following contents:
   - *net/wireless/cfg80211.ko*
   - *drivers/net/wireless/broadcom/brcm80211/brcmutil/brcmutil.ko*
   - *drivers/net/wireless/broadcom/brcm80211/brcfmac/cyw/brcfmac-cyw.ko*
   - *drivers/net/wireless/broadcom/brcm80211/brcmfmac/brcmfmac.ko*

   b) For 6.1.36 and below - from the following contents:
   - *net/wireless/cfg80211.ko*
   - *drivers/net/wireless/broadcom/brcm80211/brcmutil/brcmutil.ko*
   - *drivers/net/wireless/broadcom/brcm80211/brcmfmac/brcmfmac.ko*

   c) Backport – 5.15.32 from backport build location.
2. Copy the *cyfmacXXXX-sdio.** files in the */lib/firmware/cypress/* directory. For example, for CYW55573, create a copy as *cyfmac55573-sdio.**.
3. Run the following commands to change the access permissions.

```
chmod 777 *.sh
chmod 777 wpa_*
chmod 777 hostapd_*
chmod 777 wl*
```

4. For 6.1.36 and below, load the drivers from the respective folder,

```
root@imx8mnea-ucom:~/wifi $ rmmod brcmfmac cfg80211 compat brcmutil
root@imx8mnea-ucom:~ /wifi $ insmod ./brcmutil.ko
root@imx8mnea-ucom:~/wifi $ insmod ./compat.ko
root@imx8mnea-ucom:~ /wifi $ insmod ./cfg80211.ko
root@imx8mnea-ucom:~ /wifi $insmod ./brcmfmac.ko
```

For 6.6.23/6.12.3, load the drivers from the respective folder,

```
root@imx8mnea-ucom:~/wifi $ rmmod brcmfmac-cyw.ko brcmfmac cfg80211 compat brcmutil
root@imx8mnea-ucom:~ /wifi $ insmod ./brcmutil.ko
root@imx8mnea-ucom:~/wifi $ insmod ./compat.ko
root@imx8mnea-ucom:~ /wifi $ insmod ./cfg80211.ko
root@imx8mnea-ucom:~ /wifi $insmod ./brcmfmac.ko
root@imx8mnea-ucom:~ /wifi $insmod ./brcmfmac-cyw.ko
```

5. To verify the driver and firmware, use the `ifconfig` command. The new wireless interface name is provided in the list. For example, the following figure shows `wlan0` is the wireless interface name.

Wi-Fi setup

```
root@imx8mnea-ucom:~/CYW55573/Wi-Fi# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:1A:F1:10:69:CE
          inet addr:10.3.86.239  Bcast:10.3.86.255  Mask:255.255.255.0
          inet6 addr: fe80::21a:f1ff:fe10:69ce/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:108691 errors:0 dropped:9 overruns:0 frame:0
          TX packets:3771 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:10346382 (9.8 MiB)  TX bytes:322319 (314.7 KiB)

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:17 errors:0 dropped:0 overruns:0 frame:0
          TX packets:17 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3225 (3.1 KiB)  TX bytes:3225 (3.1 KiB)

wlan0     Link encap:Ethernet  HWaddr 00:90:4C:2D:80:01
          UP 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)

root@imx8mnea-ucom:~/CYW55573/Wi-Fi#
```

Figure 14 Verify driver and firmware

8.2 AIROC™ Wi-Fi chip as STA

Use the *wpa_supplicant* user-space Linux daemon to handle the WLAN authentication, association, disassociation, and deauthentication processes for STAs. The *wpa_supplicant* is a cross-platform supplicant providing support for WEP, WPA, WPA2, WPA3 (IEEE 802.11i), and WPA-EAP. It implements the key negotiation with an authenticator and controls the roaming and association of STA devices. *wpa_supplicant* version 2.10 or above is required. The default path for the WPA supplicant configuration file is */etc/wpa_supplicant.conf*.

Use the following Linux SystemD commands to control the operation of the WPA supplicant:

1. Initialize the *wpa_supplicant* tool using the following command:

```
./wpa_supplicant -B -c <location of .conf file > -i <interface name>
```

   For example,

```
root@imx8mnea-ucom:~ /CYW55513/Wi-Fi#  ./wpa_supplicant -B -
c/etc/wpa_supplicant.conf -iwlan0
```

2. Initialize the *wpa_cli* using the following command. This *wpa_cli* command can be used to interact with *wpa_supplicant*. It can be used to query the current status, change configuration, trigger events, and request interactive user input.

```
root@imx8mnea-ucom:~/CYW55573/Wi-Fi# ./wpa_cli
```

Wi-Fi setup

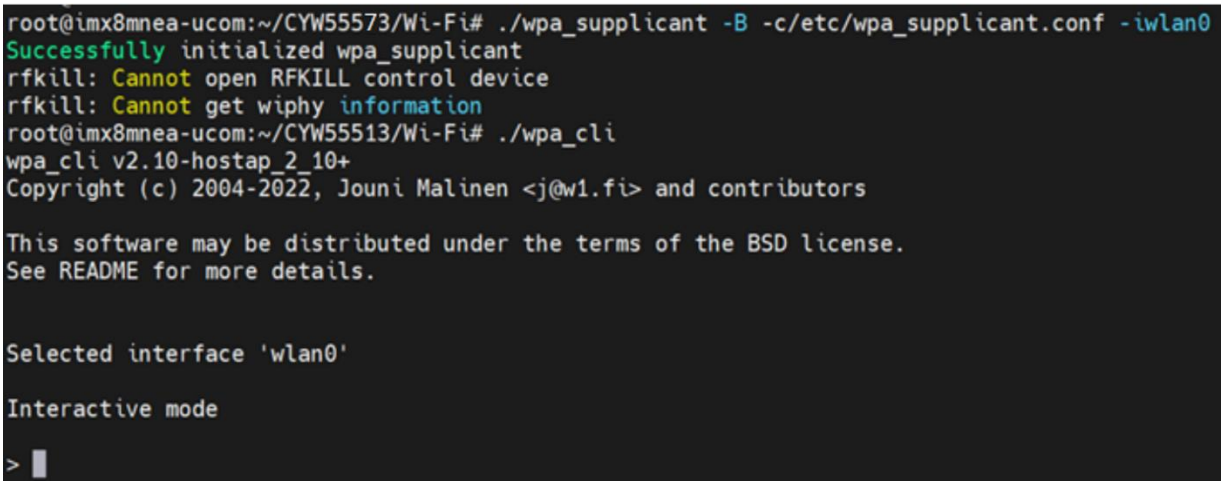

Figure 15 Initializing the wpa_cli

3. Scan for available Wi-Fi networks using the following commands:

```
scan
scan_results
```

4. Use the following command to connect to an open Wi-Fi Network.

```
remove_n all
add_network
set_network 0 ssid "ifx-open"
set_network 0 key_mgmt NONE
enable_network 0
select_network 0
status
```

Wi-Fi setup

```
> remove_n all
OK
<3>CTRL-EVENT-DISCONNECTED bssid=a6:f8:2e:88:85:b9 reason=3 locally_generated=1
<3>CTRL-EVENT-DSCP-POLICY clear_all
<3>CTRL-EVENT-NETWORK-REMOVED 0
<3>CTRL-EVENT-REGDOM-CHANGE init=CORE type=WORLD
> add_network
0
<3>CTRL-EVENT-NETWORK-ADDED 0
> set_network 0 ssid "ifx-open"
OK
> set_network 0 key_mgmt NONE
OK
> enable_network 0
OK
<3>CTRL-EVENT-SCAN-STARTED
<3>CTRL-EVENT-SCAN-RESULTS
<3>WPS-AP-AVAILABLE
<3>Trying to associate with SSID 'ifx-open'
<3>Associated with a6:f8:2e:88:85:b9
<3>CTRL-EVENT-CONNECTED - Connection to a6:f8:2e:88:85:b9 completed [id=0 id_str=]
<3>CTRL-EVENT-SUBNET-STATUS-UPDATE status=0
<3>CTRL-EVENT-REGDOM-CHANGE init=COUNTRY_IE type=COUNTRY alpha2=IN
> select_network 0
OK
> status
bssid=a6:f8:2e:88:85:b9
freq=5745
ssid=ifx-open
id=0
mode=station
pairwise_cipher=NONE
group_cipher=NONE
key_mgmt=NONE
wpa_state=COMPLETED
ip_address=192.168.30.210
p2p_device_address=02:90:4c:2e:c3:05
address=00:90:4c:2e:c3:05
uuid=4aa80dac-4a1a-5310-86ab-e2f3e6fd6aba
ieee80211ac=1
>
```

Figure 16 Connecting to an open network

5. Use the following command to connect to an authenticated SSID with wpa2-psk security:

```
remove_n all
add_network
set_network 0 ssid "ifx-wpa2"
set_network 0 proto WPA2
set_network 0 key_mgmt WPA-PSK
set_network 0 pairwise CCMP
set_network 0 psk "password"
enable_network 0
select_network 0
```

Wi-Fi setup

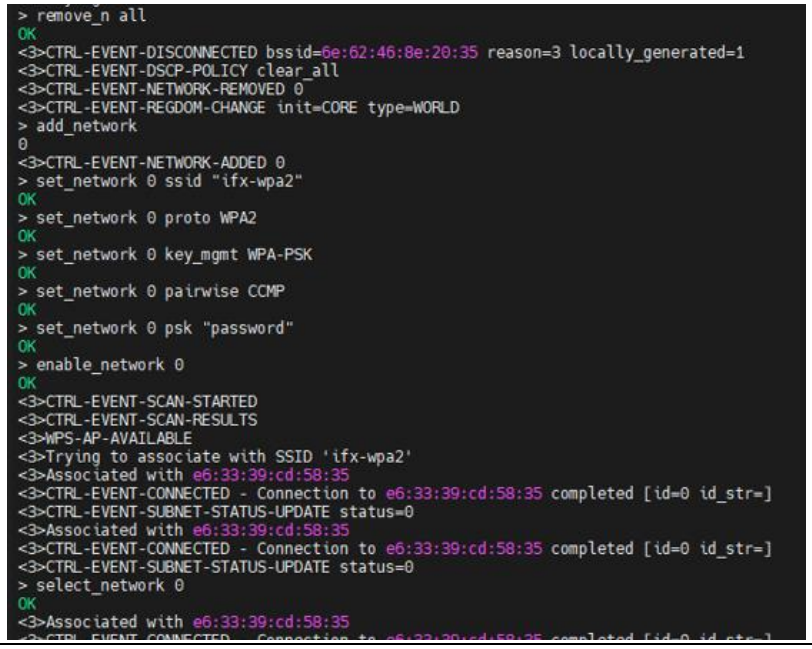

Figure 17 Connecting to an authenticated SSID with wpa2-psk security

6. Use the following command to connect to an authenticated SSID with wpa3-psk security:

```
remove_n all
scan
scan_results
add_network
set_network 0 ssid "ifx-wpa3"
set_network 0 proto WPA2
set_network 0 key_mgmt SAE
set_network 0 ieee80211w 2
set_network 0 pairwise CCMP
set_network 0 sae_password "password"
enable_network 0
```

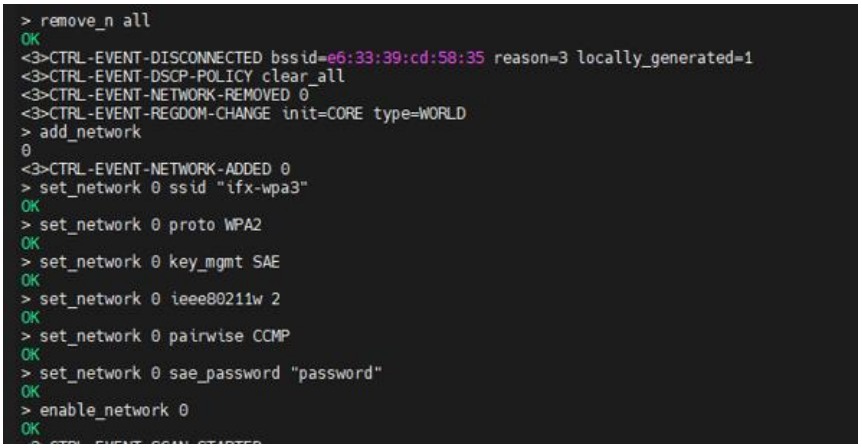

Figure 18 Connecting to an authenticated SSID with wpa3-psk security

For more information, see wpa_supplicant.

8.3 AIROC™ Wi-Fi chip as SoftAP

8.3.1 Hostapd

hostapd is a user-space daemon for setting up access points or authentication servers with fine granular control over most of the parameters. It implements IEEE 802.11 Access Point Management, IEEE 802.1X/WPA/WPA2/WPA3/EAP authenticators, a RADIUS client, an EAP server, and a RADIUS authentication server. Configure *hostapd* to function in any of those modes. Originally designed to be a daemon program, *hostapd* supports frontend programs like *hostapd_cli*.

1. Use the following command to start *hostapd*:

```
./hostapd -B <your_conf_file.conf> -dd
```

2. Use the following sample of *.conf* files for open security encryption:

```
# Hostapd configuration file for open security 5G 20MHz.
interface=wlan0
driver=nl80211
ctrl_interface=/var/run/hostapd
ssid=ifx-open
hw_mode=a
channel=40
ignore_broadcast_ssid=0
wmm_enabled=1
ieee80211ac=1
ht_capab=[SHORT-GI-20]
```

3. Use the following sample of *.conf* files for wpa2-psk encryption:

```
# Hostapd configuration file for WPA2-PSK security 5G 20MHz.
interface=wlan0
driver=nl80211
ctrl_interface=/var/run/hostapd
ssid=ifx-wpa2psk
hw_mode=a
channel=40
ignore_broadcast_ssid=0
wmm_enabled=1
ieee80211ac=1
ht_capab=[SHORT-GI-20]
wpa=2
rsn_pairwise=CCMP
wpa_passphrase=pass
```

Wi-Fi setup

4. Use the following sample of *.conf* files for wpa3-sae encryption:

```
# Hostapd configuration file for WPA3-SAE 5G 20MHz.
interface=wlan0
driver=nl80211
ctrl_interface=/var/run/hostapd
ssid=ifx-wpa3
hw_mode=a
channel=40
ignore_broadcast_ssid=0
wmm_enabled=1
ieee80211ac=1
ht_capab=[SHORT-GI-20]
wpa=2
wpa_key_mgmt=SAE
rsn_pairwise=CCMP
ieee80211w=2
sae_password=password
sae_pwe=2
```

For more information, see hostapd.

8.4 Wi-Fi: Throughput test

iPerf is a tool for active measurements of the maximum achievable bandwidth on IP networks. It supports tuning of various parameters related to timing, buffers, and protocols (TCP, UDP, and SCTP with IPv4/IPv6). For each test, it reports the bandwidth, loss, and other parameters.

Copy the iPerf tool for iMX8 from the package to */usr/sbin* and change the permissions using the following command:

```
chmod +rwx /usr/sbin/iperf
```

Throughput tests can be done using two WLAN devices.

1. Configure Device 1 as a SoftAP by following the instructions mentioned in the AIROC™ Wi-Fi chip as SoftAP and assign the *wlan0* interface to the following static IP:

```
sudo ifconfig 192.168.1.2/24
```

2. Configure Device 2 as A STA by following the instructions mentioned in AIROC™ Wi-Fi chip as STA and assign the *wlan0* interface to the following static IP:

```
sudo ifconfig 192.168.1.3/24
```

3. Ping each other:
   - From Device 1: `ping 192.168.1.3`
   - From Device 2: `ping 192.168.1.2`

Table 3 TCP throughput testing commands

Device-1-TCP server (hosted on SoftAP interface)	`iperf -s -f m -i 1 -w 8m -l 8k`
Device-2-TCP client (hosted on STA interface)	`iperf -c <Server's WLAN IP address> -i 1 -f m -t 30 -w 8m -l 8k`

Wi-Fi setup

Note: Adjust the TCP window size or try setting it to 256K (that is, `-w 256k` in the iPerf TCP server and client commands) when you see the throughput dropping to zero.

Table 4 UDP throughput testing commands

Device-1-UDP Server (hosted on SoftAP interface)	`iperf -s -u -i 1`
Device-2-UDP Client (hosted on STA interface)	`iperf -c <Server's WLAN IP address> -u -i 1 -f m -t 30 -b 200m`

Note: To get optimum performance numbers, apply the following tuning parameters:

```
# TCP/UDP buffer tuning parameters.
echo 983040 > /proc/sys/net/core/rmem_max
echo 983040 > /proc/sys/net/core/wmem_max
echo 983040 > /proc/sys/net/core/rmem_default
echo 983040 > /proc/sys/net/core/wmem_default
echo '4096 262144 983040' > /proc/sys/net/ipv4/tcp_rmem
echo '4096 262144 983040' > /proc/sys/net/ipv4/tcp_wmem
echo '122880 122880 122880' > /proc/sys/net/ipv4/tcp_mem
echo 262144 > /proc/sys/net/ipv4/tcp_limit_output_bytes
echo bic > /proc/sys/net/ipv4/tcp_congestion_control
echo 1 > /proc/sys/net/ipv4/route/flush
echo 0 >/proc/sys/net/ipv4/tcp_sack
echo 0 >/proc/sys/net/ipv4/tcp_dsack
echo 0 >/proc/sys/net/ipv4/tcp_fack
echo 0 >/proc/sys/net/ipv4/tcp_slow_start_after_idle
echo performance > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
echo performance > /sys/devices/system/cpu/cpu1/cpufreq/scaling_governor
echo performance > /sys/devices/system/cpu/cpu2/cpufreq/scaling_governor
echo performance > /sys/devices/system/cpu/cpu3/cpufreq/scaling_governor
```

Note: To improve the performance of iPerf3, run the helper script `/opt/ea/optimize_for_iperf3.sh`:

For more information, see iPerf.

9 Bluetooth®

9.1 Bring up CYW5557x for Bluetooth®

1. Use the following command to verify the BT_REG_ON pin GPIO status (see Figure 19):

```
$ gpioinfo 5
```

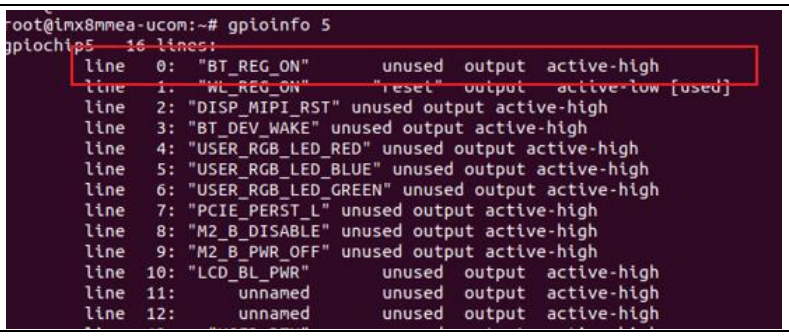

Figure 19 BT_REG_ON pin GPIO status

9.2 Application software support for Bluetooth® development

The following are the software layers:

- Application layer – code examples
- AIROC™ BTSTACK
- BTSTACK porting layer
- Manufacturing Bluetooth® Test tool (MBT)

9.2.1 Application layer – code examples

The application layer contains code examples with the basic structure that demonstrates how to use the AIROC™ BTSTACK APIs to use the Bluetooth® functionality of the AIROC™ CYW5557x combo chips. The Linux Bluetooth® code example has both Bluetooth® classic and Bluetooth® Low Energy code examples.

The following code examples are available on GitHub:

- AIROC™ BTSTACK: Bluetooth® headset for Linux host
- AIROC™ BTSTACK: Bluetooth® WakeOnLE for Linux host
- AIROC™ BTSTACK: Bluetooth® Wi-Fi Onboarding for Linux host

See the *README.md* file present in each code example directory for information on how to compile and use the code example.

Bluetooth®

9.2.2 AIROC™ BTSTACK

AIROC™ BTSTACK is optimized to work with Infineon controllers. It implements the WICED Bluetooth® APIs and supports Bluetooth® BR/EDR and Bluetooth® LE core protocols. AIROC™ BTSTACK is built with two variants: Bluetooth® LE only and Dual-Mode. The Bluetooth® LE only library has APIs that only support Bluetooth® LE features, while the Dual-Mode library supports both Bluetooth® LE and Bluetooth® classic features.

AIROC™ BTSTACK is used as a library, which can be pulled in to build a specific Bluetooth® demo or customer applications. AIROC™ BTSTACK works with the WICED library and porting layer.

For more information on AIROC™ BTSTACK, see the following:

- AIROC™ BTSTACK release notes
- Bluetooth® LE API reference manual
- Dual-Mode API reference manual

AIROC™ BTSTACK and AIROC™ BTSTACK porting layer on the GitHub repository:

- AIROC™ BTSTACK
- AIROC™ BTSTACK porting layer

9.2.3 BTSTACK porting layer

AIROC™ BTSTACK runs on a Linux host, while Bluetooth® firmware runs on Bluetooth® controller (AIROC™ combo chip). To communicate with each other, the host and controller use the HCI protocol. The AIROC™ BTSTACK library integrates with the porting layer. The porting layer implements the OS functionalities required by the stack.

The porting layer has the following functionalities:

- Firmware programing
- HCI communication with CYW5557x combo chip
- Timer
- GPIO operations

The porting layer recognizes the packets by the HCI-packet type indicator (PTI) field. From the Bluetooth® core specification, the HCI protocol supports five types of packets:

- Command
- Asynchronous Data
- Synchronous Data
- Event
- ISO Data

The PTI is a 1-byte value that precedes the HCI packet.

The transmitting path (Tx) packets can be directly sent to the controller by the AIROC™ BTSTACK. The receiving path (Rx) packets are received from the Bluetooth® thread and transmitted directly to AIROC™ BTSTACK.

HCI command packets are sent to the controller by the Tx path. HCI event packets are sent from the controller by Rx path. HCI asynchronous connection link (ACL)/synchronous connection oriented (SCO)/isochronous (ISO) data packets are sent both to the Tx path and from the Rx path to the controller.

Bluetooth®

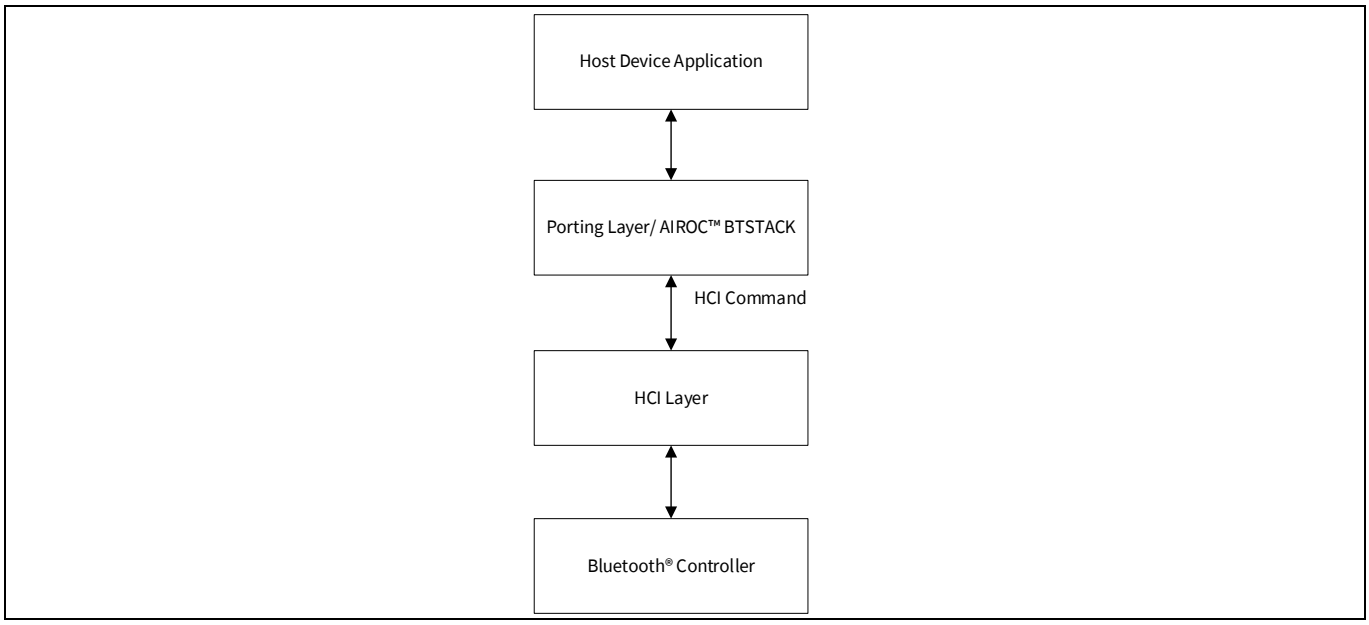

Figure 20 Software layer flow

9.3 Bluetooth® firmware

9.3.1 Bluetooth® firmware download sequence

This section describes the timing diagram and sequence involved in Bluetooth® firmware programming to RAM run by the scripts provided in Section 9.3.2.

1. Toggle VDDIO and VBAT.
2. Pull the UART CTS line of CYW5557x low and toggle the "BT_REG_ON" pin from LOW to HIGH.
3. This ensures that the device is in auto-baud mode.
4. Send the HCI Reset command to the device. This autodetects the baud rate.
5. Initiate the firmware download within 3 seconds of completing the previous step.
6. Follow the timing diagram for programming the Bluetooth® firmware to RAM.

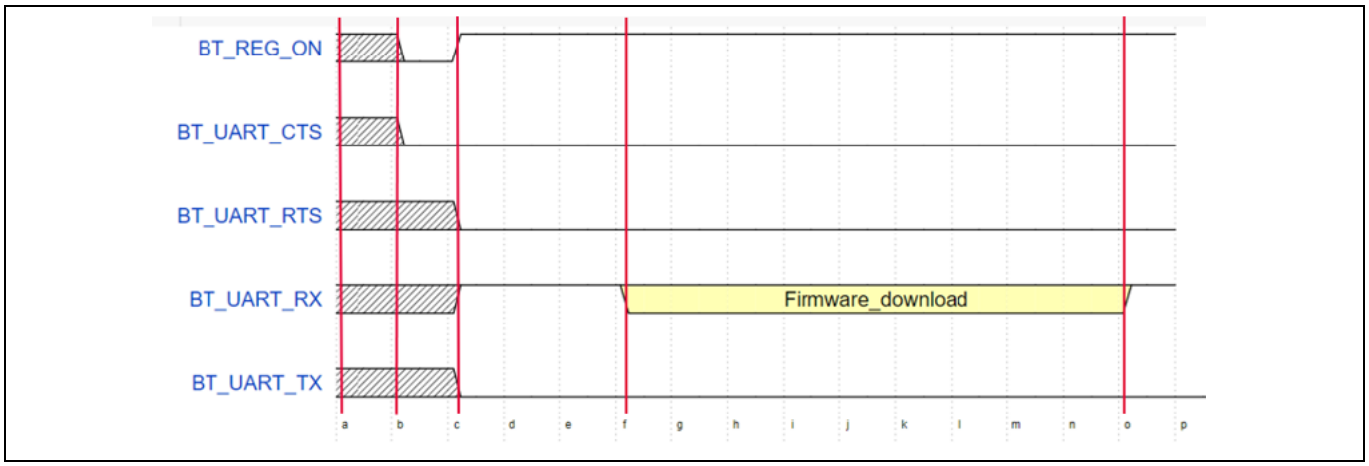

Figure 21 Timing diagram for Bluetooth® firmware programming

Bluetooth®

Do the following as indicated in Figure 21:

1. Toggle VDDIO and VBAT with a 50 ms typical value for the toggling interval.
2. Drive BT_UART_CTS from HIGH to LOW. See (b) in Figure 21.
3. Drive the BT_REG_ON from LOW to HIGH. See (c) in Figure 21.
4. Initiate the Bluetooth® firmware download within 3 seconds (as explained in Section 9.3.2) from Step 3; see (c) to (f) in Figure 21.

Note: Download the CYW5557x Bluetooth® firmware from the combo-bluetooth-firmware repo.

Bluetooth® firmware can be downloaded to the CYW5557x combo chip using the MBT tool. See Section 9.3.2.

9.3.2 Programing Bluetooth® firmware using MBT

Note: Skip this section if you are using a code example, as the code example will download the Bluetooth® firmware automatically.

1. Download the latest Manufacturing Bluetooth® Test Tool (MBT) source from the *github.com/Infineon/mbt* repo to */home/root* directory on iMX8M Nano.
2. Navigate to the */home/root/mbt/cyw5557x/scripts_for_imx8* directory that contains all necessary script files required to download the Bluetooth® firmware file.
3. Copy the Bluetooth® firmware file (CYW5557x Bluetooth® firmware) file manually to the */home/root/mbt/cyw5557x/scripts_for_imx8* directory and rename it as *"FW.hcd"*. See Antenna configuration to select the appropriate Bluetooth® firmware based on the available package.
4. Navigate to the */home/root/mbt/cyw5557x/scripts_for_imx8* directory.
5. Run the following command:

```
$ chmod +x mbt; chmod +x *.sh
```

6. To download the Bluetooth® firmware patch file, run the following autobaud command and press the **Enter** key within 3 seconds as per the following screenshot.

```
$ ./bt_load.sh
```

Bluetooth®

```
root@imx8mnea-ucom:~/mbt-main/cyw5557x/scripts_for_imx8# chmod +x mbt; chmod +x *.sh
root@imx8mnea-ucom:~/mbt-main/cyw5557x/scripts_for_imx8# ./bt_load.sh autobaud
BT_RLS_FW is:
/home/root/mbt-main/cyw5557x/scripts_for_imx8/.././
BT_TARGET_DIR:
./
BT TARGET FILE is:
FW.hcd
        <>Downloading BT FW: FW.hcd

Make chip in the autobaud mode
script:
/home/root/mbt-main/cyw5557x/scripts_for_imx8
        BT Power Off and On
REG OFF
[MBT_TRANSPORT: /dev/ttymxc0]
Wait Controller Detect CTS low
Init UART ..........
Reset the chip to autobaud state and press ENTER key!

Download BT firmware. This may take a few moments ...

Please reset the chip to autobaud state!
REG ON
rx: (7 bytes)
 04 0e 04 01 03 0c 00
Enter parse_patchram
Exit parse_patchram
Skipping minidriver download
 ... hcd files  ...
.....................................................................................................
.....................................................................................................
.....................................................................................................
.....................................................................................................
.....................................................
Exit proc_patchram 4
tx: (4 bytes)
 01 03 0c 00
rx: (7 bytes)
 04 0e 04 01 03 0c 00
Current state: Completed successfully
root@imx8mnea-ucom:~/mbt-main/cyw5557x/scripts_for_imx8#
```

Figure 22 Programing Bluetooth® firmware from iMX8M Nano script files

For more information on MBT commands and features, see the readme.

9.3.3 Antenna configuration

The CYW5557x combo chip comes with three packages: FCBGA, WLCSP, and WLBGA.

- The FCBGA package supports three different antenna configurations (Bluetooth® firmware files):
  - *fcbga_iPA_dLNA_ANT0.hcd*
  - *fcbga_iPA_sLNA_ANT0.hcd*
  - *fcbga_iPA_sLNA_BTANT.hcd*
- The WLCSP package supports three different antenna configurations (Bluetooth® firmware files):
  - *wlcsp_iPA_dLNA_ANT0.hcd*
  - *wlcsp_iPA_sLNA_ANT0.hcd*
  - *wlcsp_iPA_sLNA_BTANT.hcd*
- The WLBGA package supports three different antenna configurations (Bluetooth® firmware files):
  - *wlbga_iPA_dLNA_ANT0.hcd*
  - *wlbga_iPA_sLNA_ANT0.hcd*
  - *wlbga_iPA_sLNA_BTANT.hcd*

Bluetooth®

9.3.3.1 Hardware rework for dLNA_BTANT antenna configuration

Note: This rework applies to WLBGA and FCBGA packages only. Contact muRata for WLCSP antenna configuration.

Do the following to perform the hardware rework (in CYW5557x board) to use dLNA_BTANT antenna configurations where R49 and R50 are both 'zero' ohms.

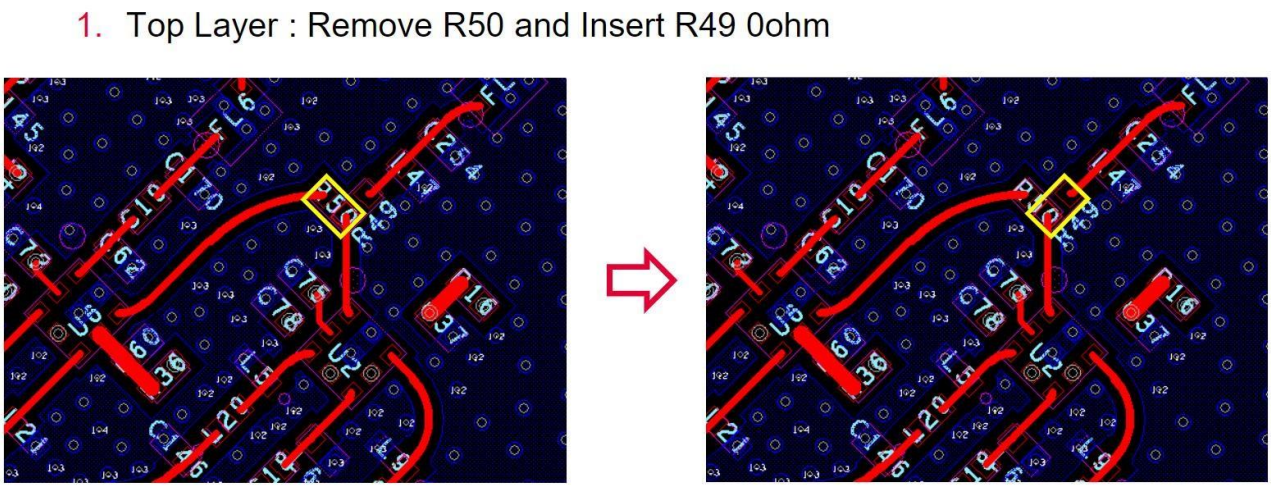

Figure 23 Hardware rework for dLNA_BTANT antenna configuration

9.4 BlueZ support

1. Download the Bluetooth® firmware patch file by following Programing Bluetooth® firmware using MBT.
2. Run the following command to set up BlueZ on a particular HCI port (*/dev/ttymxc0*):

```
$  hciattach -s 115200 /dev/ttymxc0 bcm43xx
```

3. Run the following command to configure the port:

```
$  hciconfig hci0 up
```

```
root@imx8mnea-ucom:~/mbt-main/cyw5557x/scripts_for_imx8# hciattach -s 115200 /dev/ttymxc0 bcm43xx
bcm43xx_init
Patch not found, continue anyway
Set Controller UART speed to 3000000 bit/s
Setting TTY to N_HCI line discipline
Device setup complete
root@imx8mnea-ucom:~/mbt-main/cyw5557x/scripts_for_imx8#
```

Figure 24 HCI UART enabling with *hciattach* command

4. To test the basic BlueZ stack functionality, run the following commands:

```
$ bluetoothctl
$ power on
$ agent on
$ scan on
```

Bluetooth®

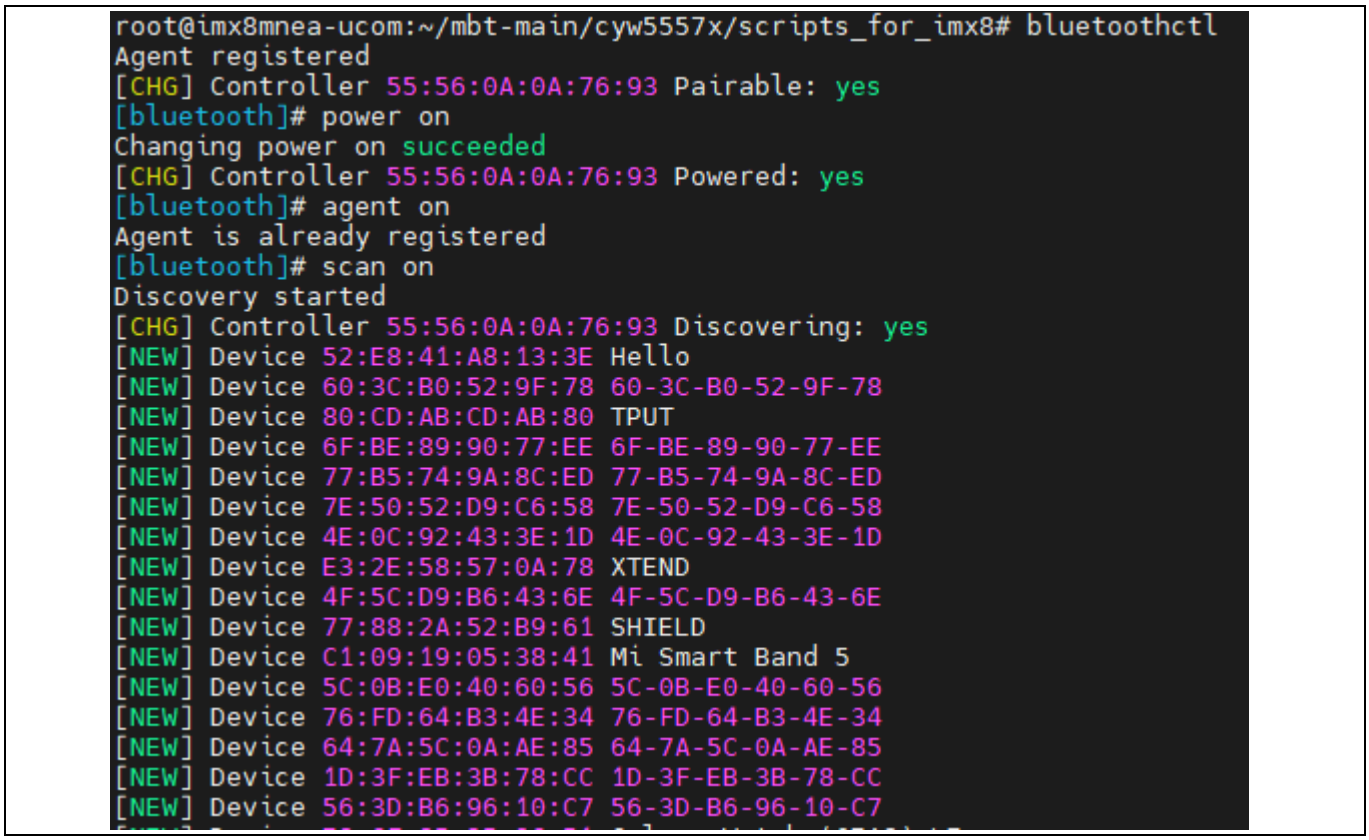

Figure 25 Sanity Bluetooth® test using BlueZ

10 Troubleshooting

10.1 Serial driver issue

If your FTDI cannot be recognized by your computer, update the FTDI USB to serial driver to fix the issue.

Locate the driver for Windows from the Drivers webpage, download, and install the driver.

10.2 How to apply the patch

To apply the patch, on copying the FMAC patches from the community, please run the below command from the iMX8 repository.

```
git am <Directory of patch>/*.patch
git add .
```

References

[1] CLM blob webpage

[2] Kernel modules webpage

[3] iMX Developer's Kit V3 webpage

[4] iMX8M Nano Developer's Kit V3 webpage

[5] iPerf - The ultimate speed test tool for TCP, UDP, and SCTP webpage

[6] hostapd: IEEE 802.11 AP, IEEE 802.1X/WPA/WPA2/WPA3/EAP/RADIUS Authenticator webpage

[7] wpa_supplicant webpage

Revision history

Revision history

Document revision	Date	Description of changes
**	2023-07-19	Initial release.
*A	2024-06-10	Added support for Linux kernel version 6.1.36. Added support for AIROC™ CYW43022.
*B	2024-11-20	Added support for Linux kernel version 6.6. Added support for AIROC™ CYW55513.
*C	2026-01-20	Added support for Linux kernel version 6.12.3.

Trademarks

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

The Bluetooth® word mark and logos are registered trademarks owned by Bluetooth SIG, Inc., and any use of such marks by Infineon is under license.

Edition 2026-01-20

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about this document?

Email:

erratum@infineon.com

002-38038 Rev. *C

Important Notice

Products which may also include samples and may be comprised of hardware or software or both ("Product(s)") are sold or provided and delivered by Infineon Technologies AG and its affiliates ("Infineon") subject to the terms and conditions of the frame supply contract or other written agreement(s) executed by a customer and Infineon or, in the absence of the foregoing, the applicable Sales Conditions of Infineon. General terms and conditions of a customer or deviations from applicable Sales Conditions of Infineon shall only be binding for Infineon if and to the extent Infineon has given its express written consent.

For the avoidance of doubt, Infineon disclaims all warranties of non-infringement of third-party rights and implied warranties such as warranties of fitness for a specific use/purpose or merchantability.

Infineon shall not be responsible for any information with respect to samples, the application or customer's specific use of any Product or for any examples or typical values given in this document.

The data contained in this document is exclusively intended for technically qualified and skilled customer representatives. It is the responsibility of the customer to evaluate the suitability of the Product for the intended application and the customer's specific use and to verify all relevant technical data contained in this document in the intended application and the customer's specific use. The customer is responsible for properly designing, programming, and testing the functionality and safety of the intended application, as well as complying with any legal requirements related to its use.

Unless otherwise explicitly approved by Infineon, Products may not be used in any application where a failure of the Products or any consequences of the use thereof can reasonably be expected to result in personal injury. However, the foregoing shall not prevent the customer from using any Product in such fields of use that Infineon has explicitly designed and sold it for, provided that the overall responsibility for the application lies with the customer.

Infineon expressly reserves the right to use its content for commercial text and data mining (TDM) according to applicable laws, e.g. Section 44b of the German Copyright Act (UrhG).If the Product includes security features:

Because no computing device can be absolutely secure, and despite security measures implemented in the Product, Infineon does not guarantee that the Product will be free from intrusion, data theft or loss, or other breaches ("Security Breaches"), and Infineon shall have no liability arising out of any Security Breaches.

If this document includes or references software:

The software is owned by Infineon under the intellectual property laws and treaties of the United States, Germany, and other countries worldwide. All rights reserved. Therefore, you may use the software only as provided in the software license agreement accompanying the software.

If no software license agreement applies, Infineon hereby grants you a personal, non-exclusive, non-transferable license (without the right to sublicense) under its intellectual property rights in the software (a) for software provided in source code form, to modify and reproduce the software solely for use with Infineon hardware products, only internally within your organization, and (b) to distribute the software in binary code form externally to end users, solely for use on Infineon hardware products. Any other use, reproduction, modification, translation, or compilation of the software is prohibited. For further information on the Product, technology, delivery terms and conditions, and prices, please contact your nearest Infineon office or visit https://www.infineon.com