

Miscellaneous

Table of Content

1. Where can I find the latest Murata documents to get started?	3
2. What are the technical forums I can use to resolve my issues?	4
3. Where can I get the 'wl' tool for my image and its source?	5
4. Which FMAC version should I use, and where can I get it from?	6
5. How do I bring up concurrent STA+AP mode on my EVK?	7
6. How do I test the throughput on my i.MX EVK + Murata module setup?	9
7. What is nvram and why do I need it?	11
8. What is dtb file and why do I need it?	12
9. How are Embedded Artists EVKs different from NXP EVKs? Which one should I choose?	13
10. How can I change the kernel configuration I am building following the Murata Linux Getting Started guide?	14
11. What is Bluetooth patch file and why do I need it?	15
12. If there are multiple Bluetooth patchfiles in the /etc/firmware folder, which one will be loaded by the Bluetooth driver?	16
13. What is the naming convention used in meta-murata-wireless layer releases?	17
14. What are the differences between brcmfmac and fmac drivers?	18
15. What is backporting and why (when) is it needed?	19
16. What software resources would I need to get started with my i.MX RT platform?	20
17. How to build an SDK using the MCUXpresso SDK builder?	23
18. How do I know the COM port number my i.MX RT board is assigned by the host PC?	28
19. How do I set up my i.MX RT platform to run the sample applications?	30
20. What is mfgtest?	32
21. How do I run manufacturing tests on i.MX platforms?	33
22. How do I run manufacturing tests on i.MX RT platform?	34
23. What is BT-pass-through?	37
24. How to test BT-pass-through on i.MX RT platforms?	38
25. What is SoftAP?	44
26. What is BLE?	45

27. How to run the BLE test app on i.MX RT platform?	46
28. How to run the iPerf sample application on i.MX RT platform?	50
29. How to get better throughput in iPerf on i.MX RT platform?	58
30. What is a CLM Blob file and why do I need it?	61
31. I have set the output power target for Channel 1 in the NVRAM file, but it is not taking effect. What can be wrong?	62
32. What is the difference between regdb and CLM blob in Linux?	63
33. How to set up P2P network with Murata modules?	64
34. How to check that the meta-murata-wireless layer is successfully added to my Yocto build?	66
35. How to reach Murata for wireless issues on the Murata modules?	68
36. Where can I find driver documentation for Murata modules?	69
37. Is there a product selector guide for Murata modules?	70
38. Which files do I need to overwrite / rename in the UUU files folder so that UUU can download my custom image?	71
39. I have followed the Murata Getting Started guide and the image files are created. But the image file has a .bz2 extension? How do I extract it?	72
40. How to set up the terminal emulator to connect to the i.MX platforms?	73
41. What are the common Yocto images that can work with Murata modules? What bitbake command should I use to build an image that will work with Murata modules?	74
42. What are the image files created by a Yocto build? Which one do I need?	75

1. Where can I find the latest Murata documents to get started?

The Murata [Quick Start Guide for Linux](#) use this document to quickly bring up your i.MX 6/7/8 board and understand the environment.

The [Quick Start Guide for FreeRTOS](#) use this document to bring up your i.MX RT board.

In case you are using one of the older Murata modules on the NXP platform, you might need to perform some rework on your NXP board to get it working. Refer to the [Murata hardware user manual](#).

For better understanding on how the Murata enabled NXP i.MX software works, you can find detailed information in the [Murata Linux User manual](#).

If you are/plan using the Murata uSD-M.2 adapter to connect your Murata module to the EVK, you can find the [datasheet here](#)

[Back to Table of Content](#)

2. What are the technical forums I can use to resolve my issues?

Murata actively monitors the following forums:

- [NXP / i.MX Processors forum](#) - for queries regarding the NXP i.MX 6/7/8 + Murata modules issues (required NXP account).
- [NXP / i.MX RT forum](#) - for queries regarding the NXP i.MX RT + Murata modules issues (required NXP account).
- [NXP / Murata / i.MX forum](#) (required NXP account, only for Murata customers).
- [Cypress / Linux forum](#) - for queries regarding Wi-Fi/BT issues of Murata modules (required Cypress account).
- [Cypress / Murata community forum](#) - for Murata specific WiFi / BT issues (required Cypress account, only for Murata customers).

[Back to Table of Content](#)

3. Where can I get the 'wl' tool for my image and its source?

You can get precompiled binaries tested with Murata modules on supported kernels at the following location:

- [for 32-bit systems](#)
- [for 64-bit systems](#)

If you need wl tool for any other platform, you need to contact Cypress support. The source code of wl tool is not available, as this is a Cypress proprietary tool.

[Back to Table of Content](#)

4. Which FMAC version should I use, and where can I get it from?

It is highly recommended that you use the latest FMAC version currently available (manda). You can get it from [murata-wireless github](#)

The production firmware and CLM blob files are [available here](#) (latest branch *manda*).

The nvram files are [available here](#) (latest branch *manda*).

The Bluetooth patchfiles are [available here](#) (latest branch *rocko-manda*).

[Back to Table of Content](#)

5. How do I bring up concurrent STA+AP mode on my EVK?

You will need to use both wpa_supplicant and hostapd applications to set up concurrent STA+AP on the i.MX EVK with Murata module. Before starting, make sure the Murata module supports STA+AP mode and you have wpa_supplicant and hostapd on your EVK roots.

- Open a terminal to the EVK from the host PC. Power on the EVK and let it boot into Linux.
- Edit the **/etc/wpa_supplicant.conf** file to make sure you add/modify a network block to include information about your available Wi-Fi network:

```
Network = {  
    ssid=<Your available network SSID>  
    psk=<Your available network passphrase> }
```
- Edit the **/etc/hostapd.conf** file to set up your AP network (change as per your requirements):

```
interface=wlan1 (Line 8)  
ssid=<AP SSID to be created> (Line 89)  
channel=11 (line 149)  
wpa=1 (Line 1138)  
wpa_passphrase=<Network passphrase to be used> (Line 1147)
```
- Kill existing wpa_supplicant process:

```
$ killall wpa_supplicant
```
- Create a new AP interface:

```
$ iw dev wlan0 interface add wlan1 type __ap
```
- Enable the new interface:

```
$ ifconfig wlan1 192.168.2.1 up
```

- Start wpa_supplicant with new configuration. This should connect the wlan0 interface (STA) to the available network:

```
$ cd /usr/sbin
```

```
$ wpa_supplicant -D nl80211 -i wlan0 -c /etc/wpa_supplicant.conf -B &
```

```
$ wpa_cli -p /var/run/wpa_supplicant/wlan0 enable_network 0
```

- Check that the connection is established:

```
$ wpa_cli -p /var/run/wpa_supplicant/wlan0 status
```

- Get an IP address for wlan0:

```
$ udhcpc -i wlan0 up
```

- Run hostapd to create the new wireless network:

```
$ hostapd /etc/hostapd.conf -B
```

- Both the AP and STA interface should be now up and running.

[Back to Table of Content](#)

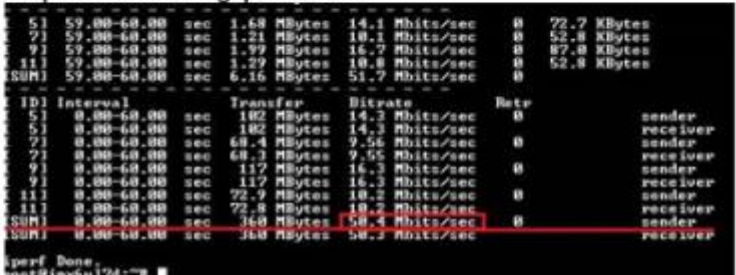
6. How do I test the throughput on my i.MX EVK + Murata module setup?

You will need an access point, and a computer which can run iperf3, connected to the access point with an Ethernet cable (Access point side). On the other side (Target side), your target EVK with Murata's WIFI/BT EVB should be connected to a Host system (which may or may not be the same computer as before). The throughput tests are performed with this setup using the commands detailed below on both sides:

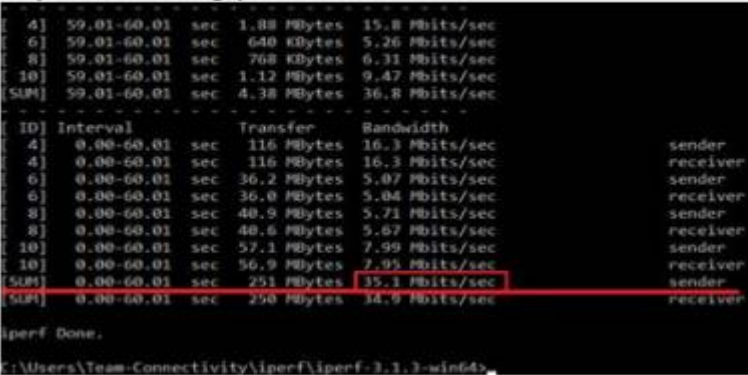
Setup:



TX throughput test:

Access point side	Target side
Step 1: Start iperf server – <code>\$ iperf3 -s -i 1</code>	Step 2: Run test – <code>\$ ifconfig wlan0 up</code> <code>\$ iw dev wlan0 connect <Your ssid></code> <code>\$ udhpcp -i wlan0</code> <code>\$ wl mpc 0</code> <code>\$ wl PM 0</code> <code>\$ wl frameburst 1</code> <code>\$ echo performance ></code> <code>/sys/devices/system/cpu/cpu0/cpufreq/scaling_governor</code> <code>\$ cat /sys/devices/system/cpu/cpu0/cpufreq/cpuinfo_cur_freq</code> <code>\$ iperf3 -c <Server's ip address> -i 1 -P 4 -t 60</code>
	Step 3: Check throughput value – 

RX throughput test:

Access point side	Target side
<p>Step 2: Run test – \$ iperf3 -c <Server's ip address> -i 1 -P 4 -t 60</p> <p>Step 3: Check throughput value –</p> 	<p>Step 1: Start iperf server – \$ iperf3 -s -i 1</p>

Command Explanations:

wl mpc 0	Set minimum power consumption mode
wl PM 0	Set driver power management mode as constantly awake
wl frameburst 1	Enable frameburst
echo performance > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor	Set the CPU frequency governor, "scaling_governor" to "performance" mode to achieve maximum performance
cat /sys/devices/system/cpu/cpu0/cpufreq/cpuinfo_cur_freq	Print current CPU speed

[Back to Table of Content](#)

7. What is nvram and why do I need it?

The NXP i.MX CPU provides a set of one-time programmable bits (e-fuses), structured in banks. These bits can be set (blown) only once. Programming these is an irreversible operation, and usually done to protect board specific information to be changed, such as the MAC address, or to prevent users from altering power control parameters. However, this is optional in production, and almost never used in development, since board parameters may need to be changed. Instead, the board parameters are kept in an editable plaintext file, called nvram file, which is loaded at board bootup. This method lets board designers tune RF components and alter critical parameters while testing boards using different versions of the nvram file. Note that the OTP memory area size is dependent on the module (e.g. Cypress CYW43012 chips have up to 356 bytes of space in OTP), so the full nvram file often cannot fit. In that case, only the board parameters that are unique to the board (such as the WLAN MC address) are written in OTP, while the rest are kept in the nvram file, even in production systems.

[Back to Table of Content](#)

8. What is dtb file and why do I need it?

DTB stands for “Device Tree Blob”. It is a binary file loaded by the bootloader during board bootup, and then passed to the Linux kernel to initialize the hardware. The DTB file contains board specific hardware configurations. The separation of kernel and board specific configuration into two separate entities allows the same kernel to be used on multiple boards, as well as multiple kernels on the same board, without needing any changes in the kernel image. A DTB file is obtained by compiling a DTS (Device Tree Source) file, and is usually supplied by the board manufacturer.

[Back to Table of Content](#)

9. How are Embedded Artists EVKs different from NXP EVKs? Which one should I choose?

EVKS from Embedded Artists and NXPs have their advantages and disadvantages. To understand which will suit your need better, refer to the comparison table below:

	NXP i.MX	Embedded Artists i.MX
Pros	<ul style="list-style-type: none"> NXP is always the first platform to come out with support for a new kernel. All MCU/MPU combinations are supported. Very good customer adoption rate and large market base – which means more active community forums. NXP engineering team focus efforts on debugging issues. Cypress supports debug efforts on NXP platforms. Newer platforms have started adopting the M.2 interface. 	<ul style="list-style-type: none"> Better documentation. Covers various “holes” not covered by NXP. Consistent M.2 interface across all platforms. RF performance optimization with support for 3.3/3.6V VBAT. Enhanced WLAN debug capabilities on dev kits (JTAG, WL UART, BT UART). Supports enhanced Cypress M.2 debug pinout. Direct plug-in support for CyBlueTool. Excellent technical collaboration with Murata.
Cons	<ul style="list-style-type: none"> Driven by hard deadlines, which sometimes limit optimal module selection choices. Some i.MX platforms come with soldered down old Murata modules that are no longer actively supported. Legacy i.MX6 platforms require unwieldy interfacing solutions for Murata modules Newer i.MX platforms use M.2, but only bring out WLAN-PCIe and not WLAN-SDIO Do not bring out WLAN/BT debug signals 	<ul style="list-style-type: none"> Not all MCU/MPU options are supported. Some COM boards have fixed 3.3V VIO on SDIO interface. This makes some Murata modules (such as 1LV) not possible to use with. Primary support focus of NXP and Cypress engineering is not Embedded Artists. Fortunately, Do not bring out WLAN/BT debug signals

[Back to Table of Content](#)

10. How can I change the kernel configuration I am building following the Murata Linux Getting Started guide?

The following bitbake command allows you to edit the kernel configurations. Make sure to execute this before the final bitbake command to build the image.

```
$ bitbake -c menuconfig linux-imx
```

Note that, changes are lost if you remove the tmp directory or do a "bitbake -c clean linux-imx".

[Back to Table of Content](#)

11. What is Bluetooth patch file and why do I need it?

Bluetooth patchfile (*.hcd) is a module specific Bluetooth firmware binary file. The file is used during Linux/BlueZ “hciattach” call to configure the Bluetooth core. Hciattach tool queries the module for its chipset number and chipset version, and searches for the required patchfile in the /etc/firmware folder.

[Back to Table of Content](#)

12. If there are multiple Bluetooth patchfiles in the /etc/firmware folder, which one will be loaded by the Bluetooth driver?

The Bluetooth patchfile to load is selected based upon two parameters obtained from the module – the chipset Id and the chipset version. The .hcd file whose name starts with the following string is selected:

“BCM” + <CYW chipset number> + <CYW chipset version>

Note that, there may be additional characters after the initial string, these are ignored. In case there are multiple .hcd file for the same chipset number and chipset ID, then the first file (alphabetically) is selected.

[Back to Table of Content](#)

13. What is the naming convention used in meta-murata-wireless layer releases?

Murata maintains multiple branches in the Murata github for meta-murata-wireless Yocto layer. The branch names follow the following convention:

<i.MX architecture>-<Linux kernel>-<FMAC version>

- i.MX architecture specifies the i.MX family – “imx” (i.MX 6/7, 32-bit) or “imx8” (i.MX 8, 64-bit).
- Linux kernel release is specified using the Yocto codename – “sumo”, “rocko”, morty”, “krogoth” etc. The latest version is “sumo”. To learn about Yocto release codenames, refer to the [LINK](#).

(Fun fact: The codenames are taken from the Total Annihilation video game franchise)

- FMAC version is specified using Cypress FMAC version codename – “manda”, “mothra”, “battrra”, “orga”. The latest one Murata supports is “manda”.

(Fun fact: The codenames are all monster names from the Godzilla universe)

For example, the latest branch for i.MX 6 platforms is “imx-sumo-manda”.

[Back to Table of Content](#)

14. What are the differences between brcmfmac and fmac drivers?

The main differences are as following:

brcmfmac	fmac
Released as part of Linux kernel.	Released directly by Cypress as tarball.
Linux community supported (not supported by Murata).	Officially supported by Cypress and Murata.
Only basic testing done before release	Fully tested.
Included WLAN/BT firmware are not always correct	Includes correct WLAN/BT firmware files.
WPA supplicant and hostapd are not patched for correct operation with Cypress chipsets.	Fully patched WPA supplicant and hostapd included.
No additional effort required to integrate with Linux kernel.	Needs to be incorporated with Linux kernel for operation. However, fully integrated with Linux backport tool for automatic porting to various kernels.

Murata highly recommends using the latest version of the FMAC driver *only*, to be used with Murata modules, and provides easy mechanism for integration. For more information, check the [Murata Quick Start guide](#).

[Back to Table of Content](#)

15. What is backporting and why (when) is it needed?

Backporting refers to a mechanism of porting a device driver to older kernels . Backporting allows the users to stay in an old kernel version and still use the latest drivers from mainline, with support of the latest features.

Since Cypress FMAC driver is released separately from the Linux kernel, backporting is important to make sure it can run on most of the current kernels. For kernels supported by Murata, scripts are provided in [Murata github](#) to automatically backport FMAC driver for the user. For manual backporting steps, refer to the [Cypress FMAC release README](#) and the [backporting blog entry](#).

Note that, from kernel versions 4.14.x and later, the FMAC version has become a part of the Linux kernel and no longer requires backporting.

[Back to Table of Content](#)

16. What software resources would I need to get started with my i.MX RT platform?

A great place to start would be the Murata Quick Start Guide for FreeRTOS, available from the [Murata Wireless landing page](#)



Wi-Fi Bluetooth for NXP i.MX

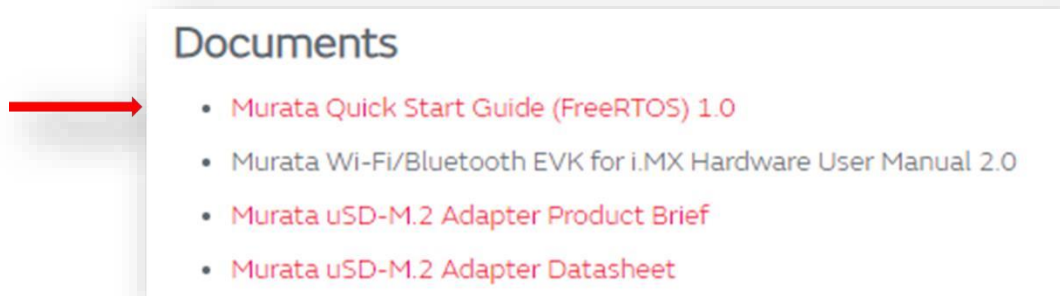
Murata has partnered with NXP Semiconductors N.V., Cypress Semiconductor Corporation, and Embedded Artists AB to offer a complete Wi-Fi and Bluetooth connectivity environment for building world class Internet-connected products. The Murata Connectivity Modules enable developers to minimize their development time and effort for connectivity function implementation. Developers can get well verified Hardware and Software for Murata Wi-Fi and Bluetooth module on NXP i.MX platforms. Both Linux and FreeRTOS are supported.

i.MX 6/7/8 series are supported with Linux and i.MX RT series are supported with FreeRTOS.

- Linux based solution
- FreeRTOS based solution

Linux Based Solution

Features	Applications
• World Class Quality Wi-Fi and Bluetooth module	• Smart devices



Documents

- [Murata Quick Start Guide \(FreeRTOS\) 1.0](#)
- [Murata Wi-Fi/Bluetooth EVK for i.MX Hardware User Manual 2.0](#)
- [Murata uSD-M.2 Adapter Product Brief](#)
- [Murata uSD-M.2 Adapter Datasheet](#)

MCUXpresso IDE is an IDE by NXP which provides the easiest way to work with i.MX RT boards. It is available for free at: <http://www.nxp.com/mcuxpresso>.

You will also need an SDK, which you can customize for your platform and download

from: <https://mcuxpresso.nxp.com>.

Quick tip: Select 'all components' when downloading the SDK. The aws_examples and wifi_examples are for Murata Modules specifically.



If you are using a Windows version earlier than Windows 10, you may also have to install the mbed serial port driver for correct detection of the i.MX RT platform. You can get it from [here](#).

You can also refer to the [NXP Getting started guides](#) (for i.MX RT 1050; you can select a different platform as well).



If you have not already got a terminal application, you should install one as well. [Tera Term](#) is a good choice.

If you are planning to test BT pass-through, you will require the Cypress CyBluetooth application. You can get it from the following link:

- For [Windows](#)
- For [Linux](#)



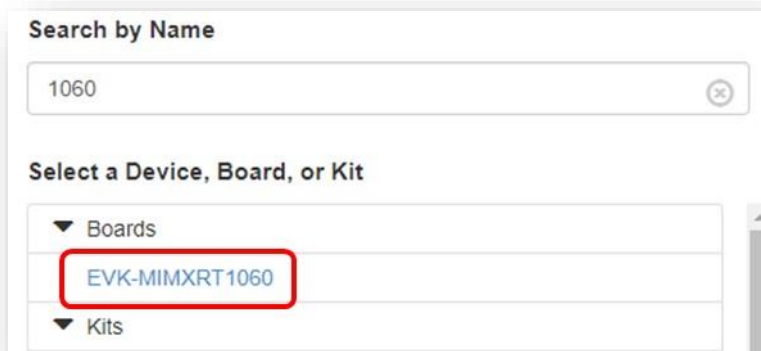
[Back to Table of Content](#)

17. How to build an SDK using the MCUXpresso SDK builder?


Step 1: Visit [MCUXpresso SDK Builder](#), then click “Select Development Board”.



Step 2: Type “1060”, then select EVK-IMXRT1060 for i.MX RT 1060 EVK. For i.MX RT 1050 EVKB, type “1050” and select “EVKB-IMXRT1050”.



Step 3: Scroll down, then click “Build MCUXpresso SDK”.



Hardware Details

Board	EVK-MIMXRT1060
Device	MIMXRT1062
Core Type / Max Freq	Cortex-M7F / 600MHz
Device Memory Size	0 KB Flash 1024 KB RAM

Actions

- [Build MCUXpresso SDK](#)
- [Explore selection with Pins tool](#)
- [Explore selection with Clocks tool](#)

Step 4: Click “Add software component”.

SDK Builder

Generate a downloadable SDK archive for use with desktop MCUXpresso Tools.

Developer Environment Settings
Selections here will impact files and examples projects included in the SDK and Generated Projects

Host OS: Windows Toolchain / IDE: MCUXpresso IDE

Select Optional Middleware
Add middleware, operating systems, and software libraries to your SDK.

+ Add software component

This MCUXpresso SDK configuration is available for direct download

Download SDK

Archive Name: SDK_2.6.2_EVK-MIMXRT1060

Don't use: <, >, ~, ", /, \, ?, *, ! in the name of your SDK

Step 5: Click on “Select All” and select “Save changes”.

Select Software Components

Select optional components from the list below to be added to your SDK.

21 items selected

Select All Deselect All

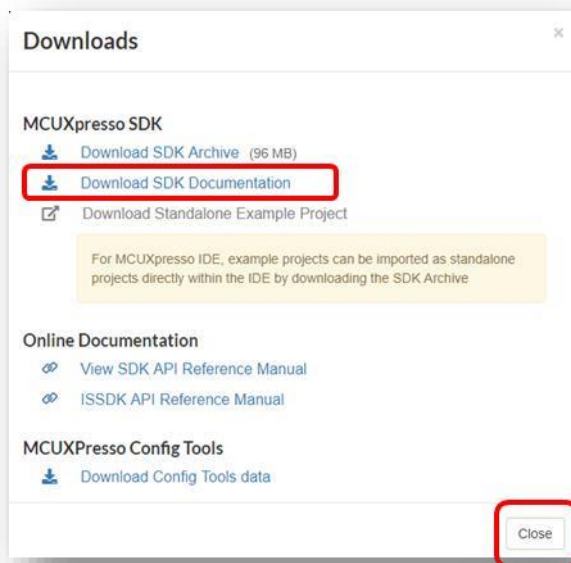
Component	Status
CMSIS DSP Library	✓
Middleware	
AWS IoT	✓
elQ	✓
emWin	✓
FatFS	✓
FreeMASTER	✓
ISSDK	✓
JPEG library	✓
littlefs	✓
lwIP	✓
mbdttls	✓

Cancel Save changes

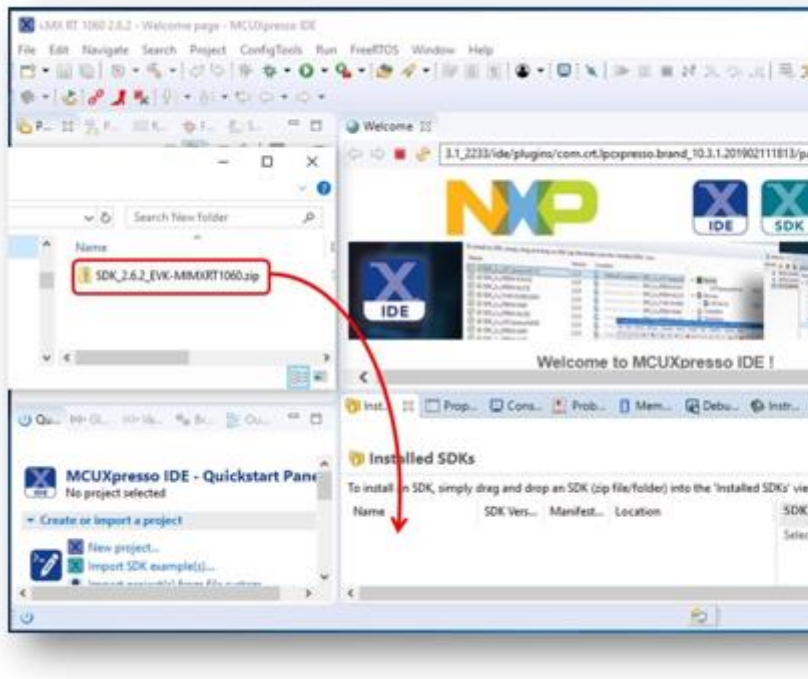
Step 6: Click “Download SDK”, then agree the EULA to download the SDK.



Step 7: Click “Download SDK Archive” if download doesn’t start automatically.



Step 8: Drag and drop the downloaded SDK zip file to “Installed SDKs window” in the MCUXpresso IDE to install.



[Back to Table of Content](#)

18. How do I know the COM port number my i.MX RT board is assigned by the host PC?

For Linux systems:

Step 1: Plug in the USB cable, connected to the i.MX RT board, to the host PC.

Step 2: Check the kernel log, the new interface should be listed:

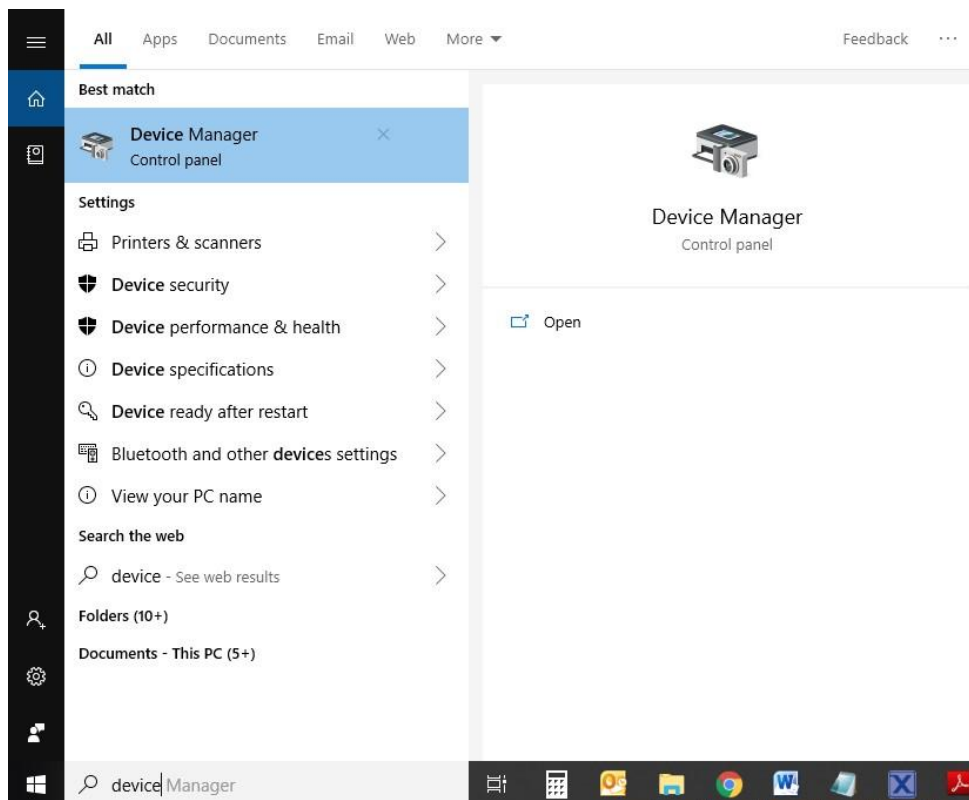
```
$ dmesg | grep "ttyUSB"
[503175.307873] usb 3-12: cp210x converter now attached to ttyUSB0
[503175.309372] usb 3-12: cp210x converter now attached to ttyUSB1
```

There should be two ports, one is the Cortex-A core debug console, and the other is for Cortex M4. You need to use the first one.

For Windows systems:

Step 1: Plug in the USB cable, connected to the i.MX RT board, to the host PC.

Step 2: Open the Windows “Device Manager” (search for “device manager” in the Start menu search bar)



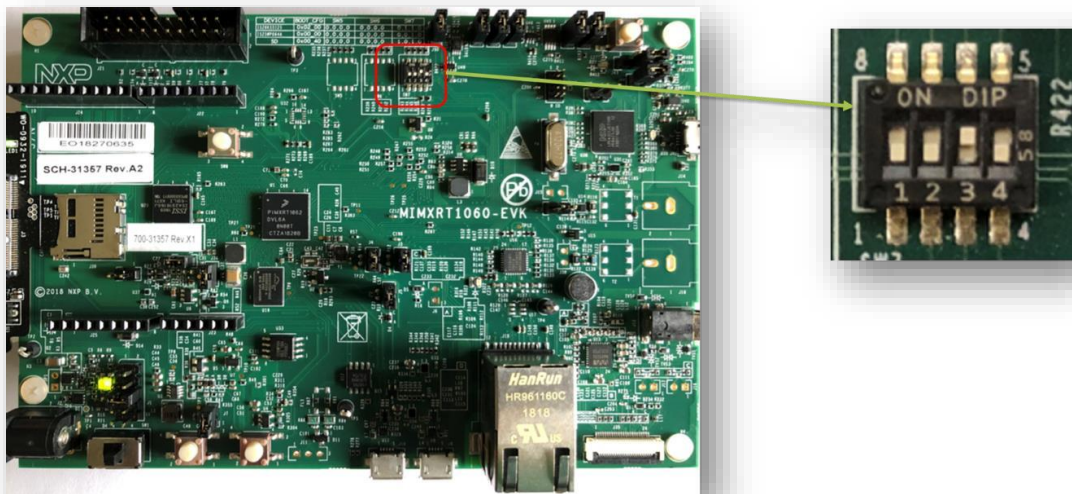
Step 3: Expand the “Ports (COM & LPT)” section to see the available COM ports. The i.MX RT board COM port should be named “mbed Serial Port” and the port number should be mentioned beside.

- > Audio inputs and outputs
- > Batteries
- > Bluetooth
- > Cameras
- > Computer
- > Disk drives
- > Display adapters
- > Firmware
- > Human Interface Devices
- > Imaging devices
- > Jungo Connectivity
- > Keyboards
- > Mice and other pointing devices
- > Monitors
- > Network adapters
- ✓ > Other devices
 - WebUSB: CMSIS-DAP
- > Portable Devices
- ✓ > Ports (COM & LPT)
 - mbed Serial Port (COM4)
- > Print queues
- > Printers
- > Processors
- > Security devices
- > Software components
- > Software devices
- > Sound, video and game controllers
- > Storage controllers
- > System devices
- > Universal Serial Bus controllers
- > WSD Print Provider

[Back to Table of Content](#)

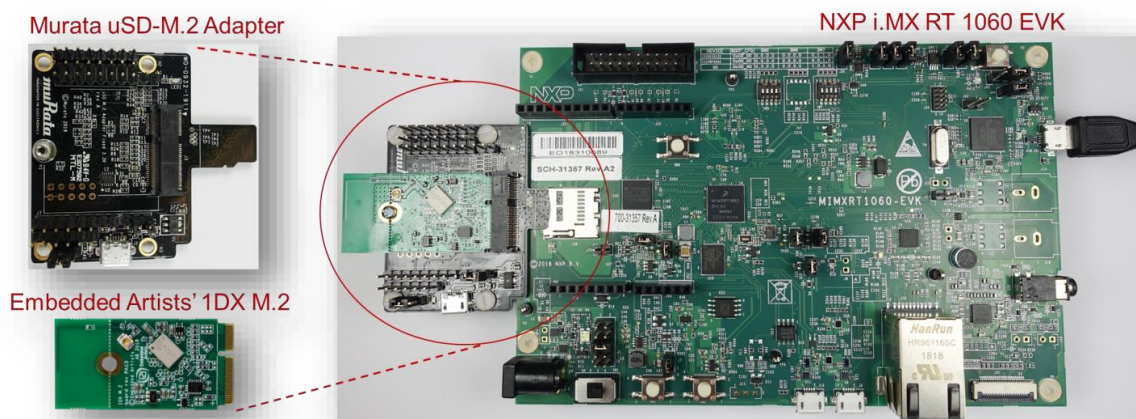
19. How do I set up my i.MX RT platform to run the sample applications?

Step 1: Before booting the board, make sure the SW7 DIP switches are set up correctly (1-OFF, 2-OFF, 3-ON, 4-OFF). This allows the board to boot from Hyperflash.

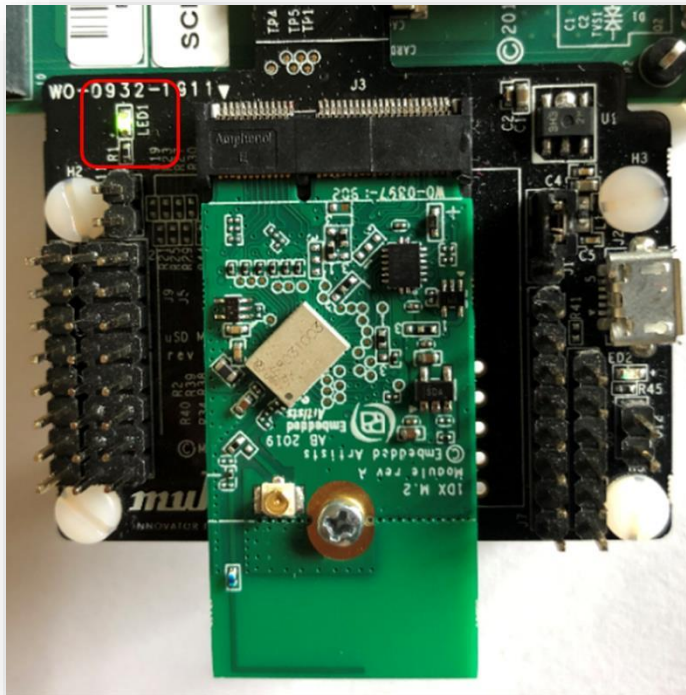


Step 2: Connect the EVK to Adapter and M.2 module.

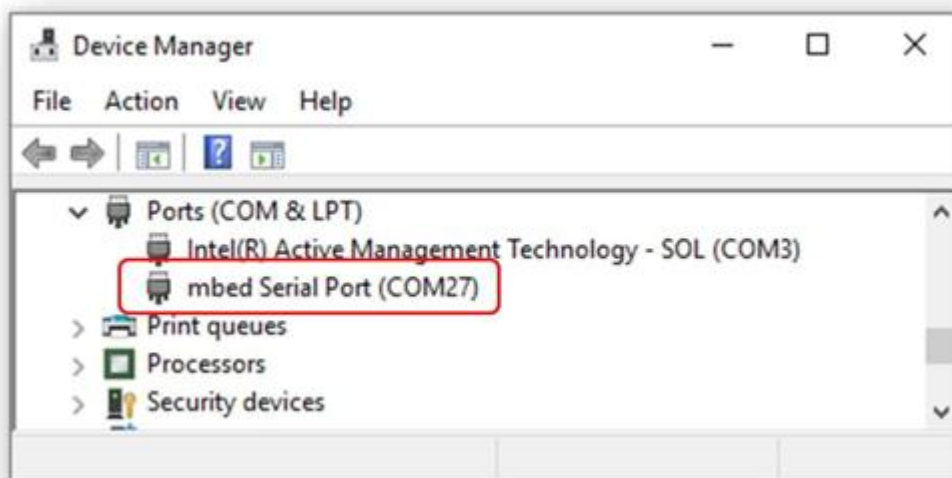
N.B. This configuration shown supports WLAN only.



Step 3: Connect the USB to the host PC and make sure the green LED (LED1) of the uSD-M.2 adapter is on.



Step 4: Open Windows Device Manager and you should be able to see mbed Serial Port if the board is detected correctly (and the software installation is successful).



[Back to Table of Content](#)

20. What is mfgtest?

Manufacturing test is a mechanism provided by the WLAN FMAC driver to perform RF / regulatory testing. This is especially required during certification. You will need to use a manufacturing test firmware instead of the regular production firmware to run manufacturing tests on the module. In addition, you will require the Cypress 'wl' application to issue test commands to the module.

[Back to Table of Content](#)

21. How do I run manufacturing tests on i.MX platforms?

Running manufacturing tests is straightforward with the FMAC driver. You will need the following:

- A manufacturing test firmware: Request [here](#) for a manufacturing test firmware and provide the production firmware version you are currently using.

```
brcmfmac: brcmf_c_preinit_dcmds: Firmware version = wl0: Sep 21  
2018 04:08:34 version 7.45.173 (r707987 CY) FWID 01-d2799ea2
```

- The 'wl' tool: Obtain the wl tool binary from Murata github for [32-bit 'wl' tool](#) (to be used in i.MX 6/7 platforms), or [64-bit 'wl' tool](#) (to be used in i.MX 8 platforms).

Once these are obtained, replace your production firmware with the manufacturing firmware (we recommend you rename the production firmware to ensure it does not get overwritten). To ensure you have the manufacturing firmware loaded, check for the firmware version in the board boot log – the string WLTEST should appear.

```
brcmfmac: brcmf_c_preinit_dcmds: Firmware version = wl0: Sep 21 2018  
04:02:19 version 7.45.173 (r707987 CY WLTEST) FWID 01-3c82dde4
```

Note that the version number of the production and manufacturing firmware should not differ. Also, manufacturing test firmware does not support some interoperability modes that production firmware does. The manufacturing test firmware is a specific release and is intended only to be used for RF testing.

Before running any 'wl' test, make sure wpa_supplicant is not running:

```
$ killall wpa_supplicant
```

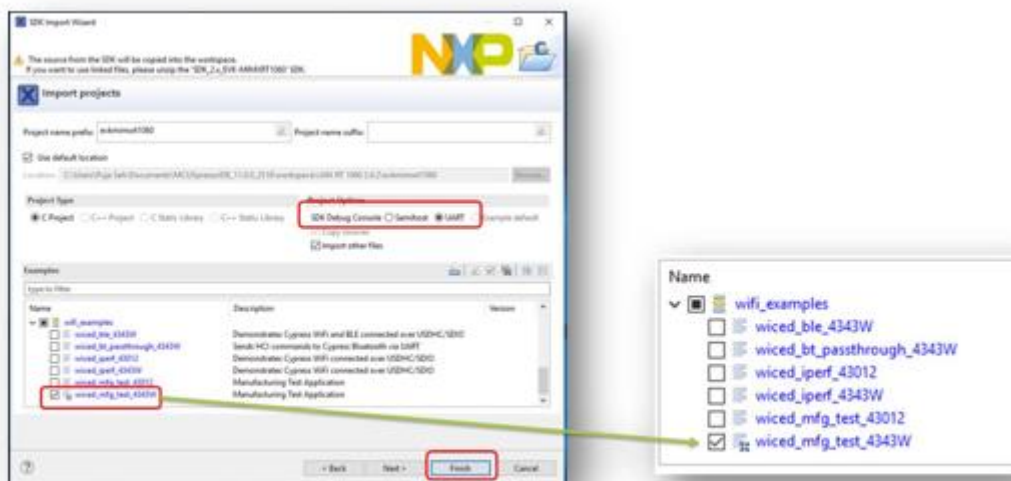
Once RF testing is done, you can revert back to regular operation by replacing the manufacturing firmware with production firmware.

[Back to Table of Content](#)

22. How do I run manufacturing tests on i.MX RT platform?

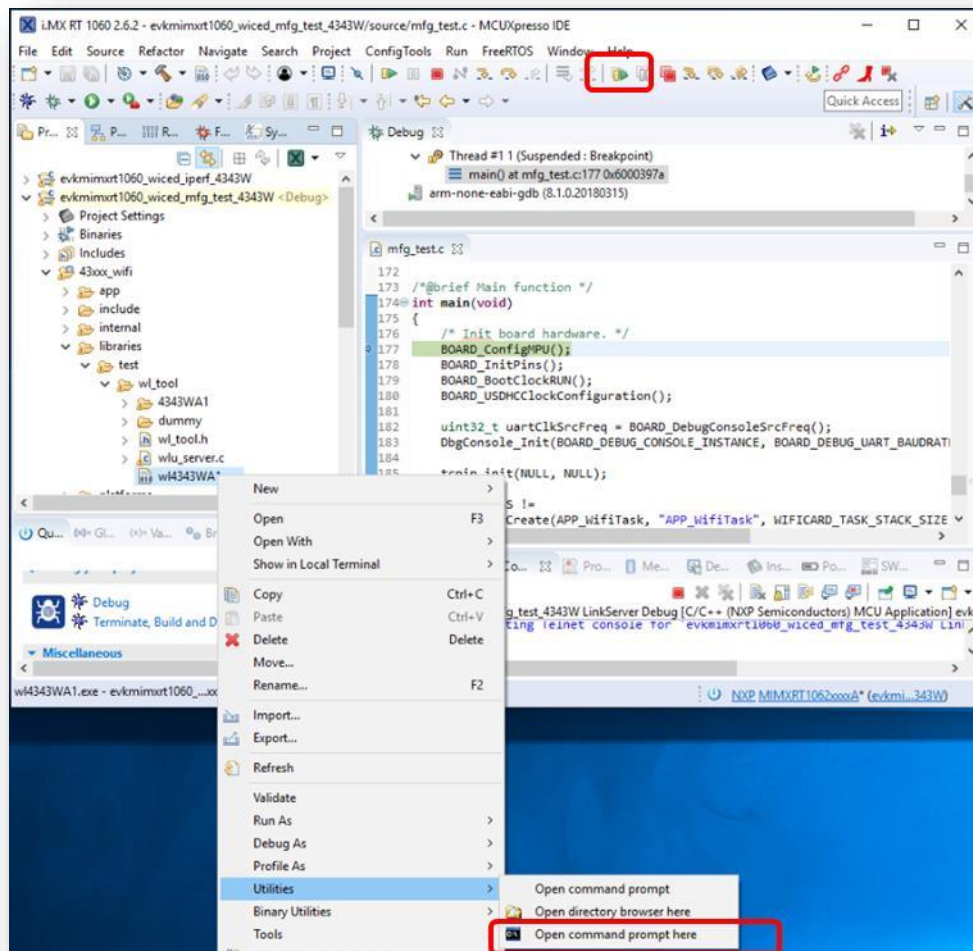
It is assumed you have already set up the MCUXpresso IDE and installed the SDK for your board. You can follow the Murata Quick Start Guide available [on Murata Wireless page](#). The following steps use the RT1050 EVK + 1DX EVB as example.

Step 1: Click on Import SDK example and select wiced_mfg_test_4343W to run MFG test.

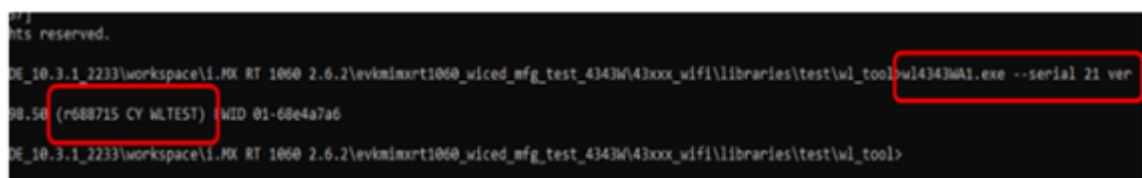


Step 2: Run the debug in the IDE. No need to run the IDE in release mode. No changes in code are required.

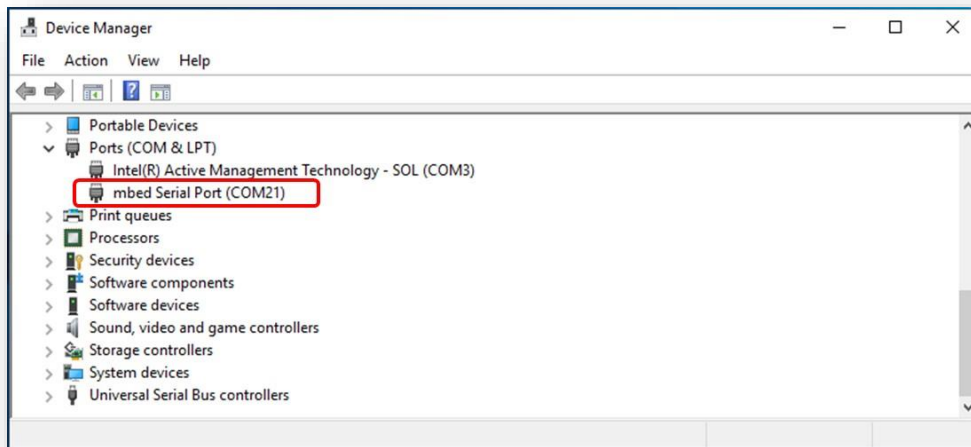
Step 3: Right click on 43xx_wifi/libraries/test/wl_tool/wl4343WA1.exe and open a command prompt. Resume debug.



Step 4: Run the command “wl4343WA1.exe --serial <COM port number> ver” on the open command prompt. The wireless firmware version information should show up. Ensure it is a manufacturing firmware – the WLTEST string should be present.



You can find the port number to use in the Windows Device Manager.



[Back to Table of Content](#)

23. What is BT-pass-through?

BT pass-through is a mechanism provided by Cypress BT modules which overtakes the Bluetooth UART communication between the Bluetooth module and the platform OS, and provides a direct communication channel to the Bluetooth module from a development machine. This communication channel can be used to control/test the Bluetooth functionalities from a personal computer using a tool capable of sending HCI commands, such as CyBlueTool.

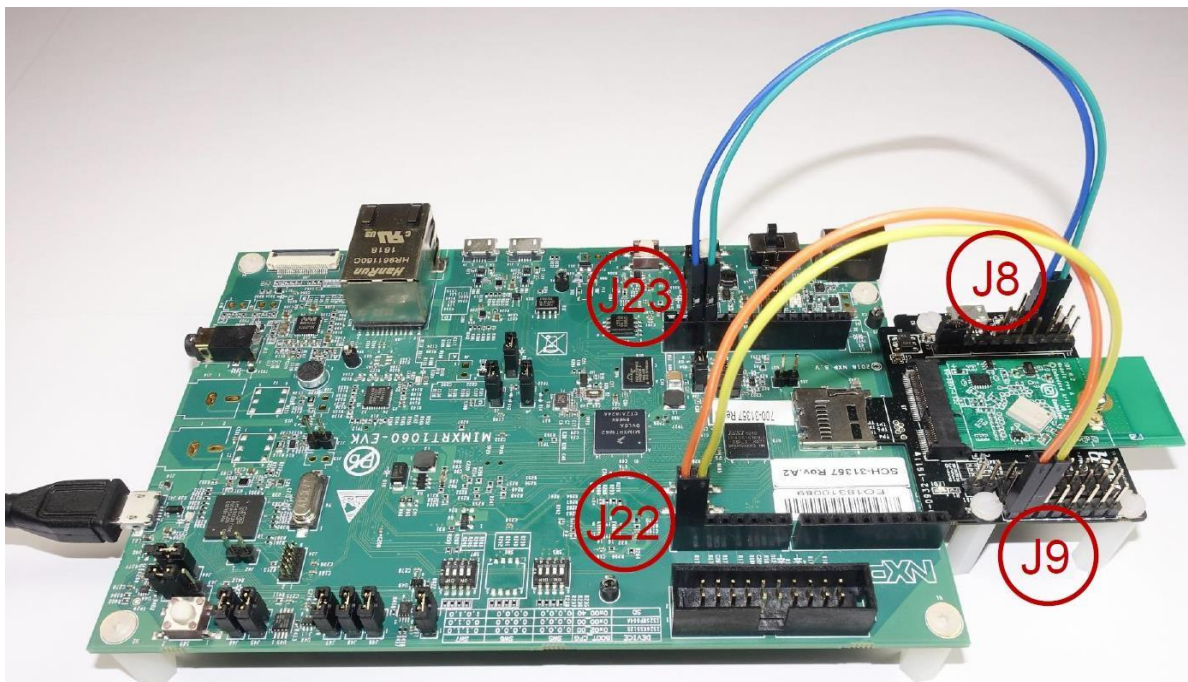
Check the FAQ question [here](#) for instructions on how to perform BT pass-through using CyBlueTool on Embedded Artists i.MX platforms.

[Back to Table of Content](#)

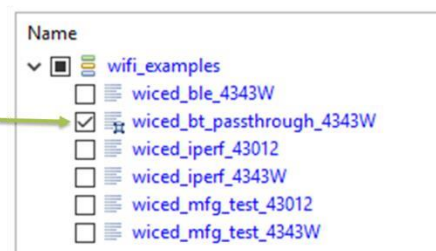
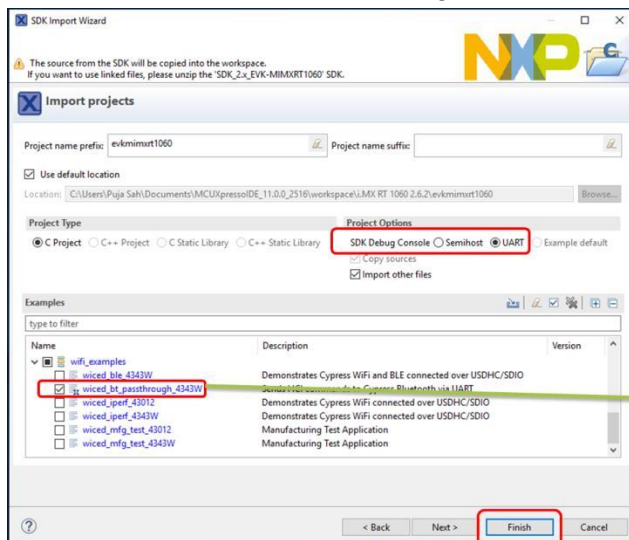
24. How to test BT-pass-through on i.MX RT platforms?

Step 1: Connect i.MX RT board J22 (pins 1-2) to uSD-M.2 adapter's J9 (pins 1-2). This intercepts the BT UART TX/RX signals.

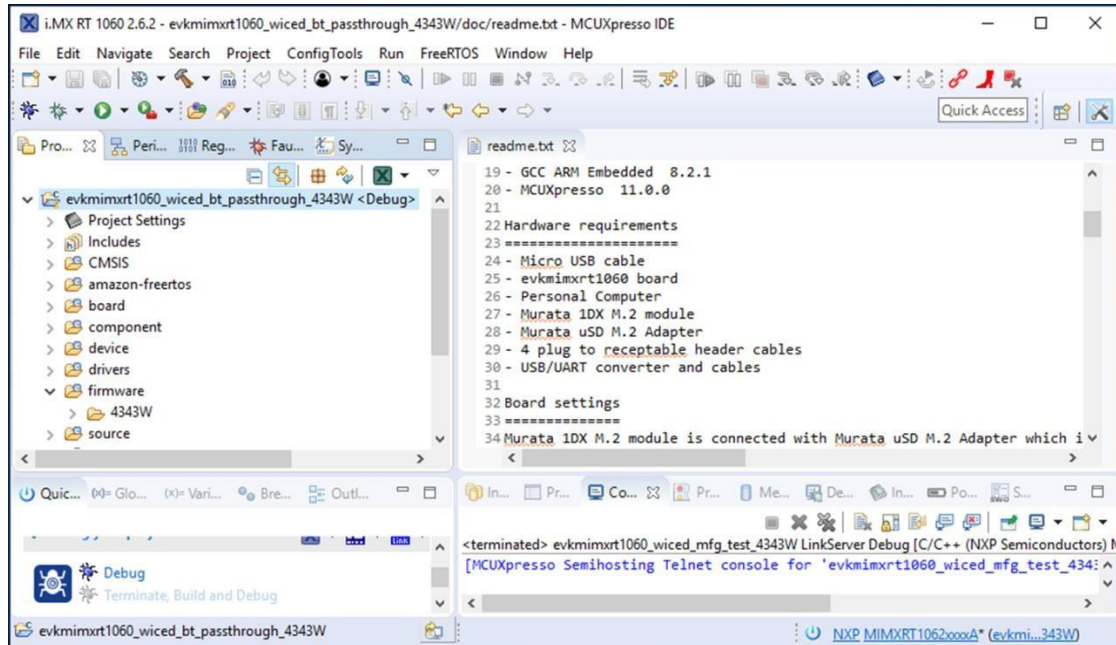
Step 2: Connect i.MX RT board J23 (pins 3-4) to uSD-M.2 adapter's J8 (pins 3-4). This intercepts the BT UART CTS/RTS signals.



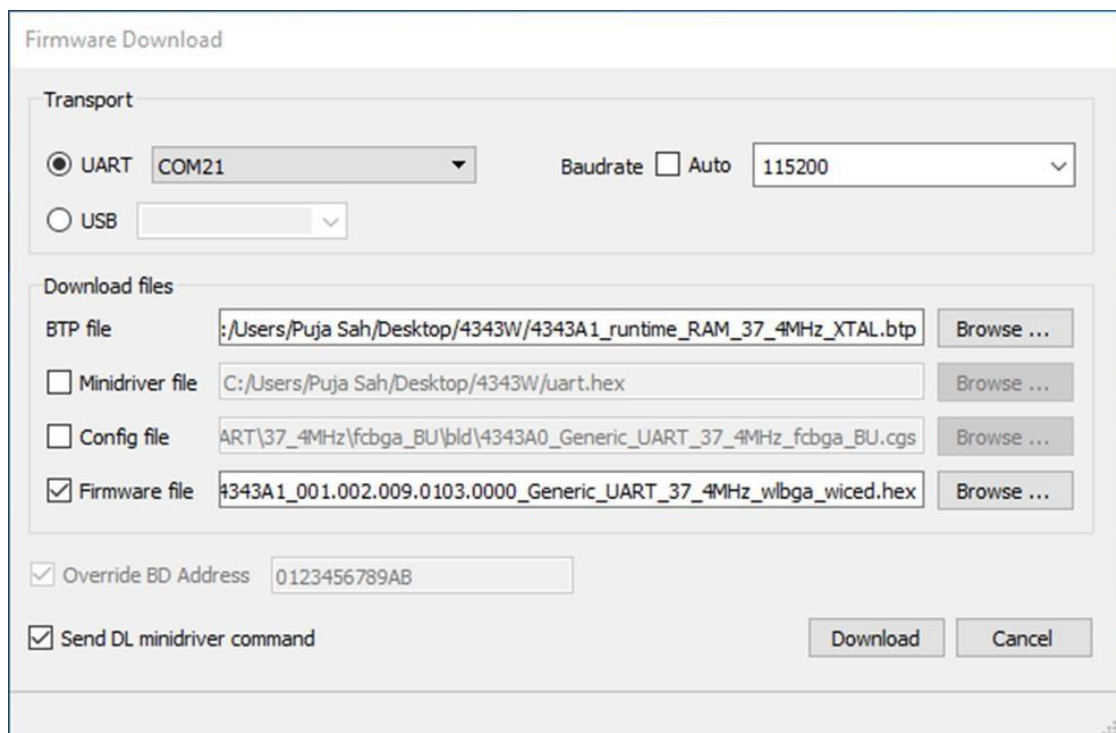
Step 3: Select the wiced_bt_passthrough_4343W example in the MCUXpresso IDE. Select UART as SDK Debug Console.



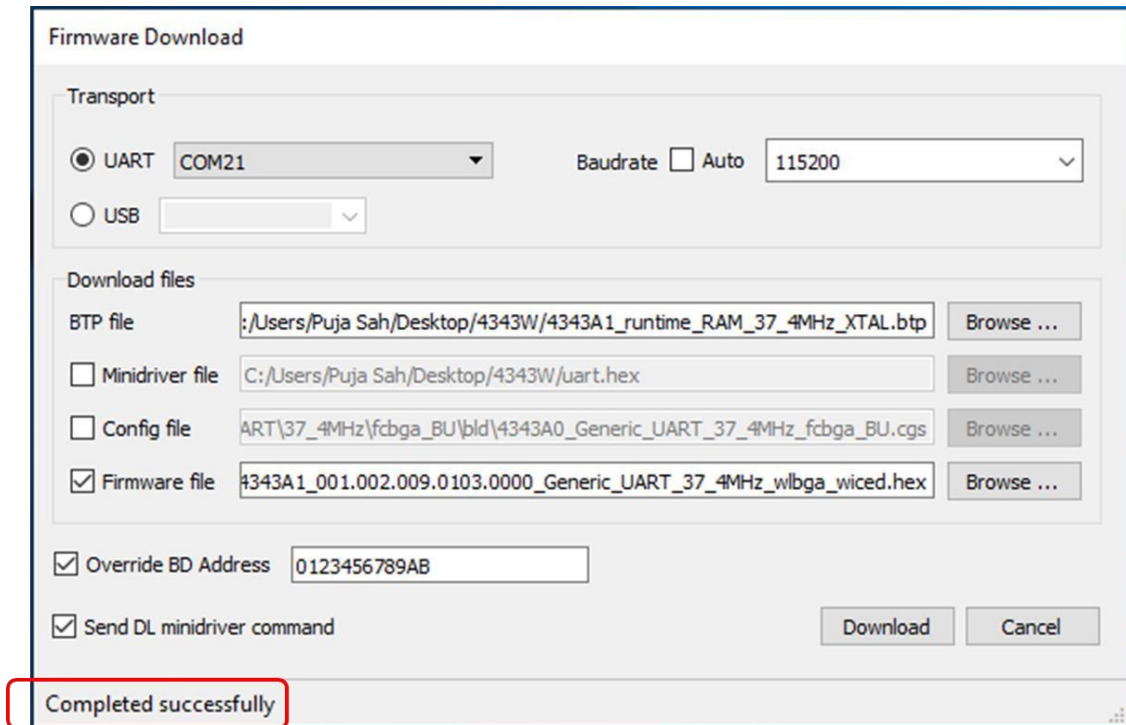
Step 4: Run the debug in the IDE.



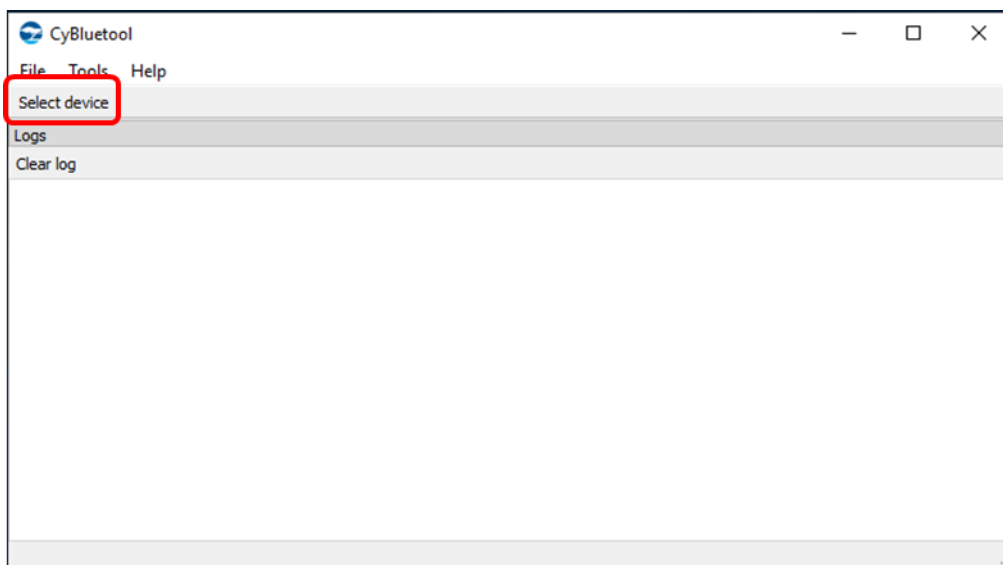
Step 5: Open CyBlutool. Go to Tools/Firmware download. Choose the mbed serial port number in UART.



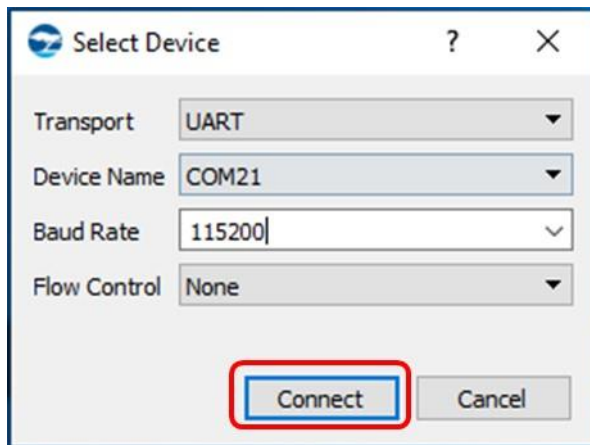
Step 6: After the firmware is downloaded successfully, the status will be updated.



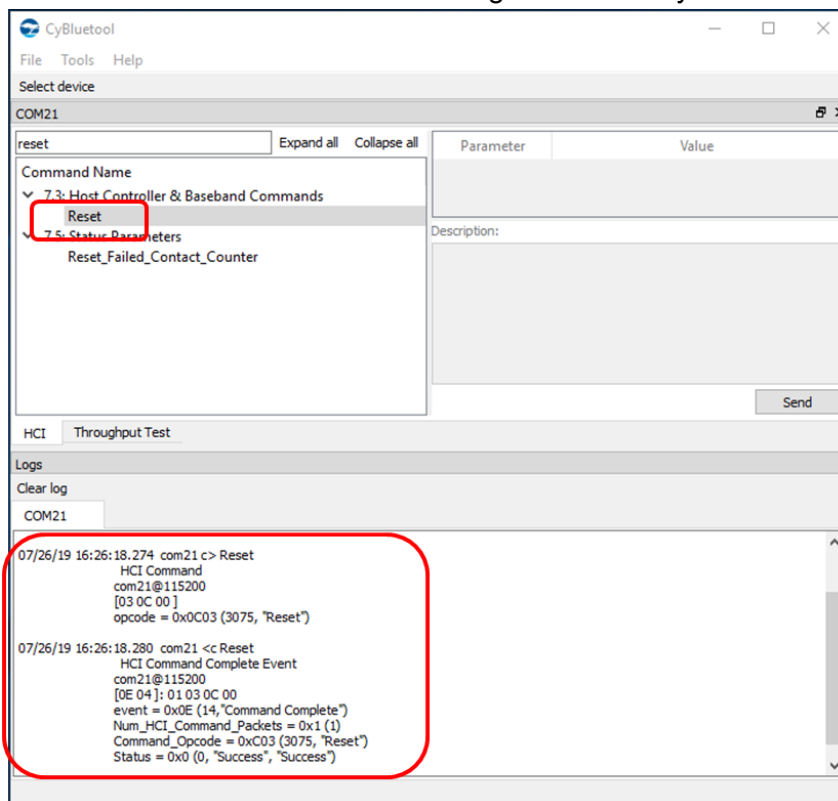
Step 7: Press "Select device" in the CyBluetool window.



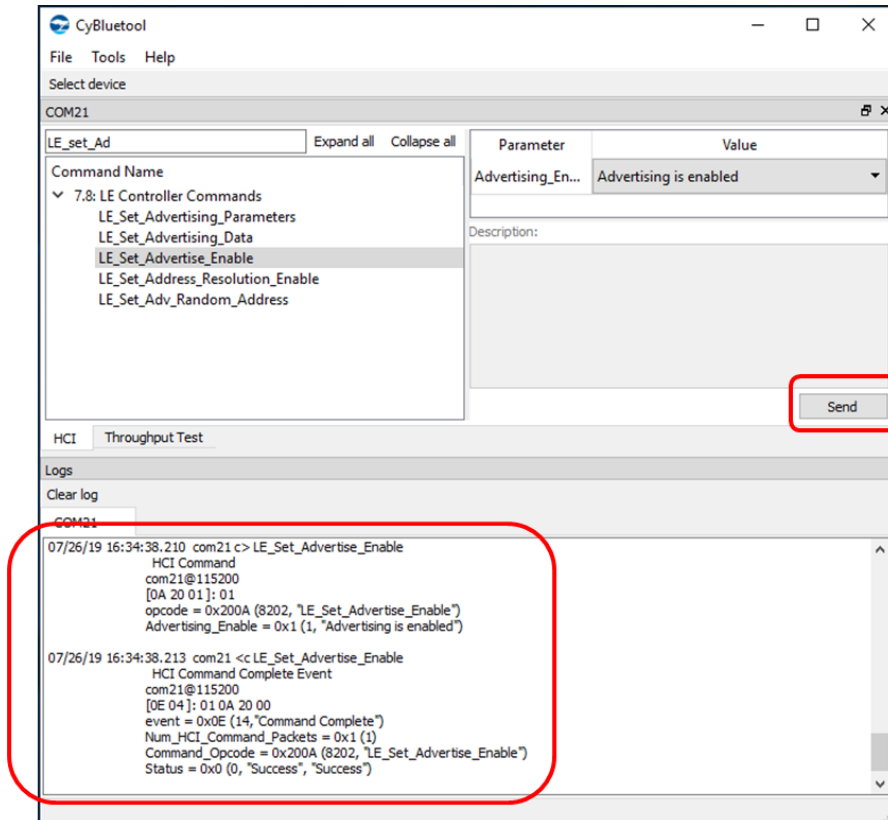
Step 8: Choose same COM port number as you chose while downloading firmware.



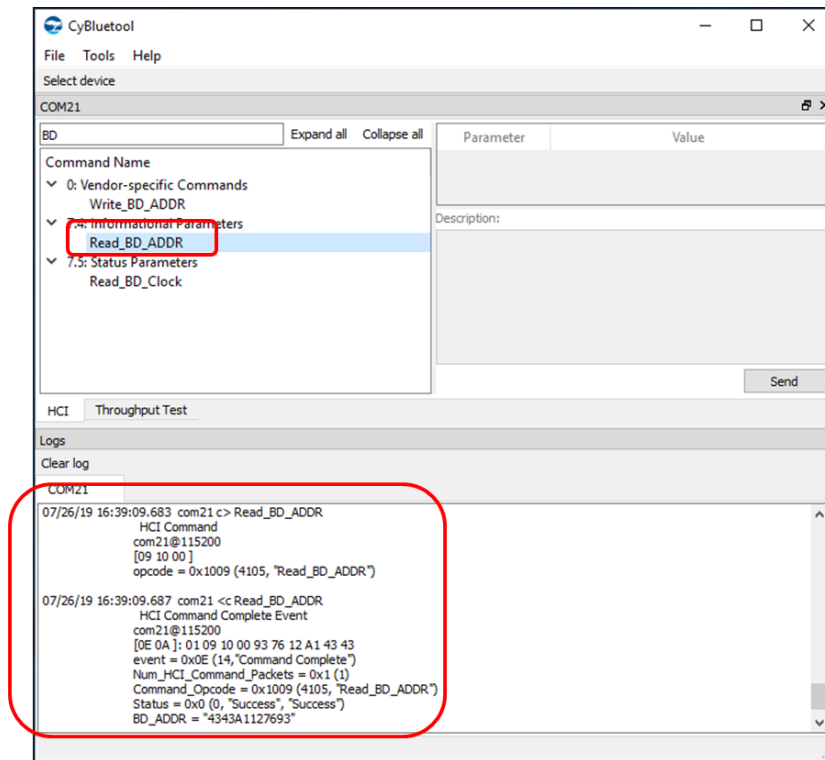
Step 9: Type reset in the search box. Double click on the reset command. You should now be able to see “success” in the log window of CyBlueTool.



Step 10: You can also test the HCI command LE_Set_Advertise_Enable with parameter Advertising_Enable set to "Advertising is enabled" in CyBluetool.



Step 11: The HCI command Read_BD_ADDR in CyBluetool gives you the BD address.



[Back to Table of Content](#)

25. What is SoftAP?

SoftAP is an abbreviated term for "software enabled access point". This refers to enabling Access Point features, via software only, on a Wi-Fi module that is primarily designed to be used as a client/station. This is also known as Wi-Fi Hot Spot and is a common use case of WiFi modules. In Linux based systems, FMAC driver provides interfacing with the hostapd software to provide SoftAP feature. You will additionally need udhcpd application if you want the SoftAP to support dynamic IP address assignment. However, udhcpd is not required if you are going to use static addressing.

The hostapd application requires a configuration file (/etc/hostapd.conf), which must include all the SoftAP settings. A sample configuration file is supplied with hostapd and you can refer to it for details. In general, the common settings are:

- interface=<Wireless interface name. E.g. wlan0>
- channel=<Channel to use. E.g. 11>
- ssid=<Network SSID to create. E.g. "Test_AP">
- wpa=<1|0 – to enable or disable WPA>
- wpa_passphrase=<WPA password, if wpa is enabled>

Before running hostapd, make sure wpa_supplicant is not running:

```
$ killall wpa_supplicant
```

Now start the hostapd application and the AP interface should be up and discoverable from clients.

```
$ ifconfig wlan0 192.168.1.1 up
```

```
$ hostapd -B /etc/hostapd.conf
```

[Back to Table of Content](#)

26. What is BLE?

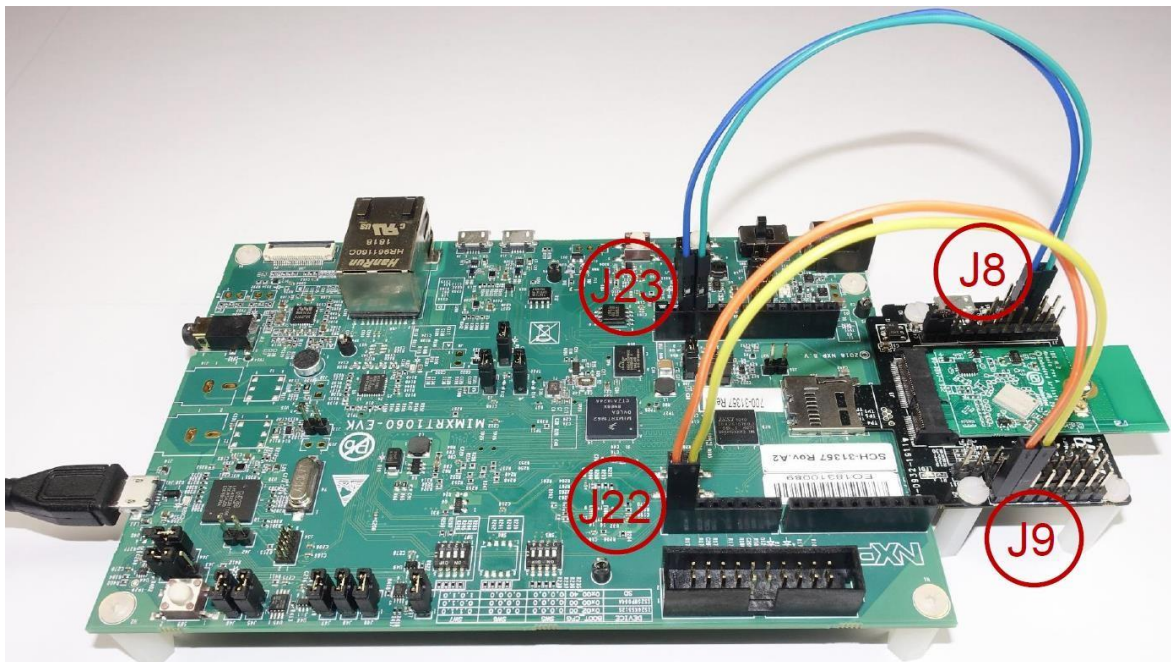
BLE stands for Bluetooth Low Energy. This refers to a low energy version of the standard Bluetooth communication (known as Bluetooth Classic), which provides considerably reduced power consumption and cost while maintaining a similar communication range. All inproduction Murata Wi-Fi+BT combo modules support BLE.

[Back to Table of Content](#)

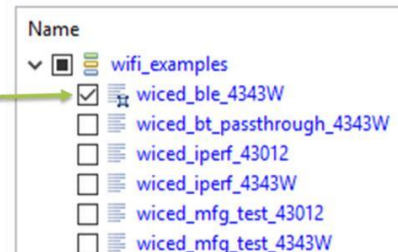
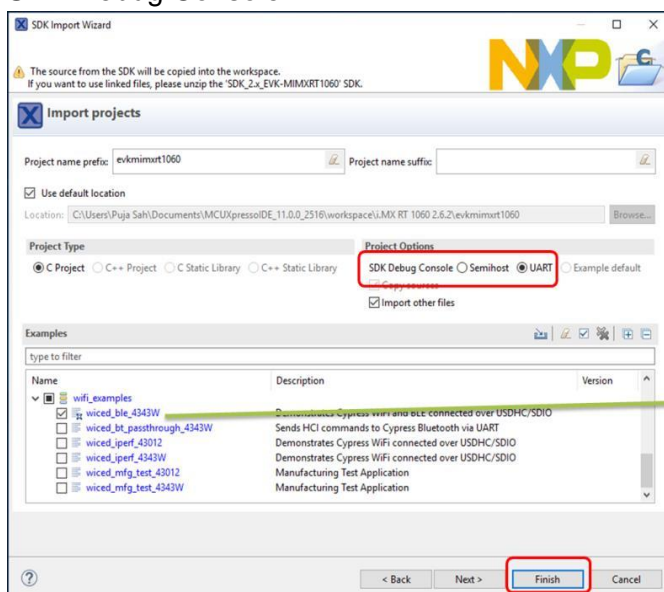
27. How to run the BLE test app on i.MX RT platform?

Step 1: Connect i.MX RT board J22 (pins 1-2) to uSD-M.2 adapter's J9 (pins 1-2). This intercepts the BT UART TX/RX signals.

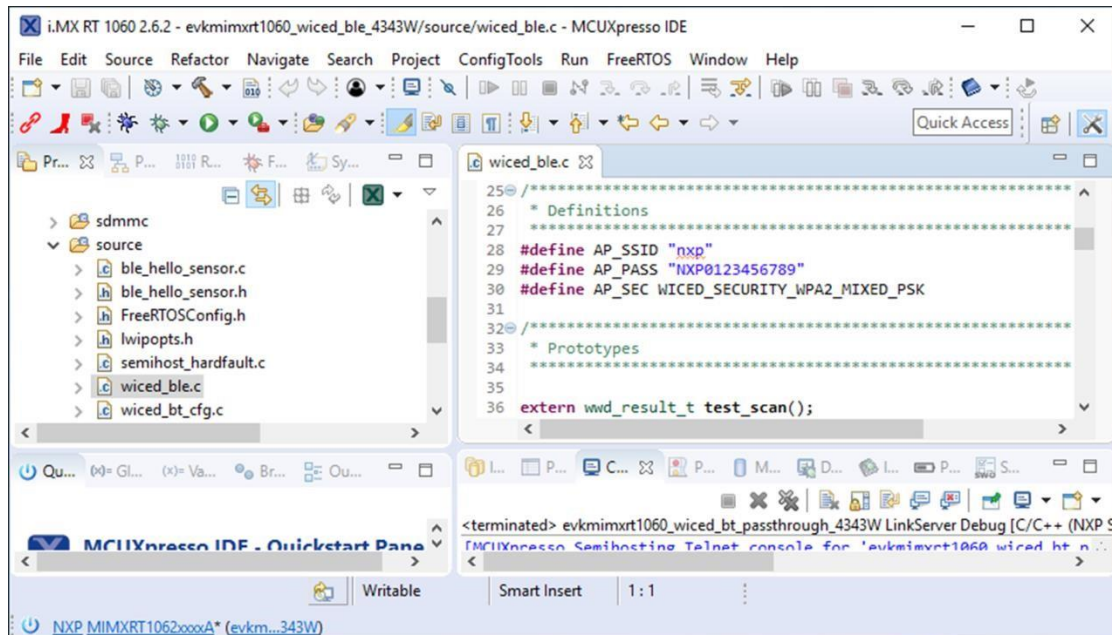
Step 2: Connect i.MX RT board J23 (pins 3-4) to uSD-M.2 adapter's J8 (pins 3-4). This intercepts the BT UART CTS/RTS signals.



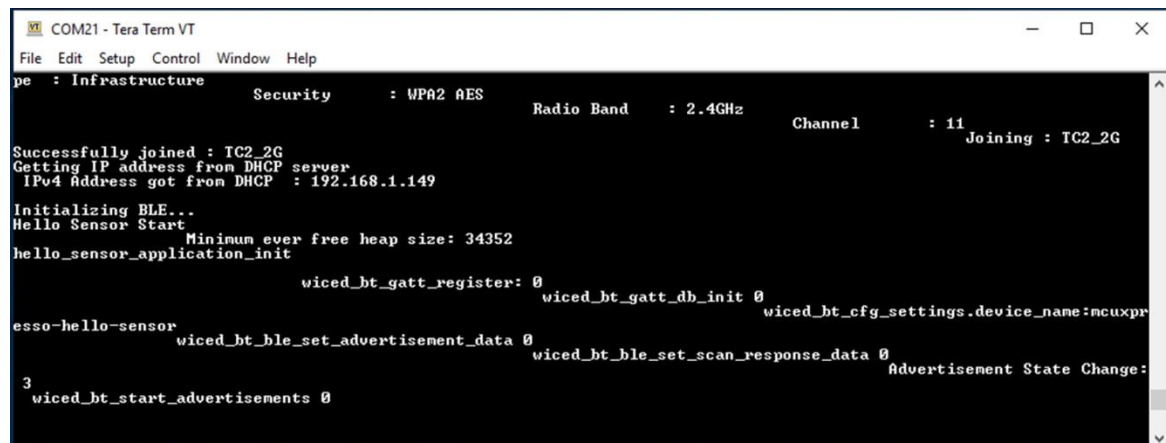
Step 3: Select the wiced_ble_4343W example in the MCUXpresso IDE. Select UART as SDK Debug Console.



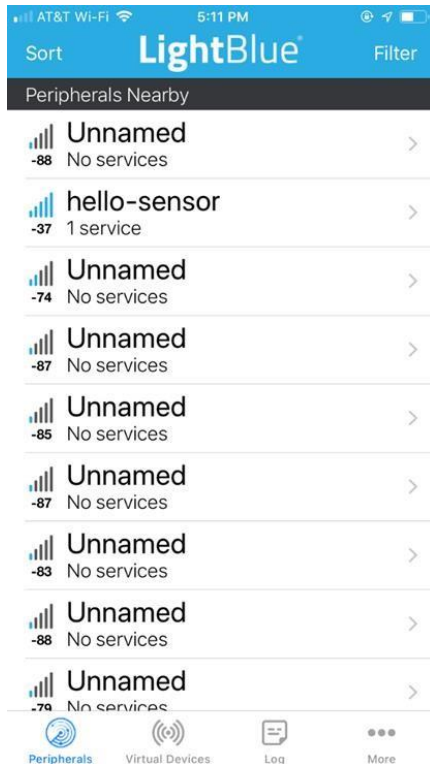
Step 4: (OPTIONAL) Go to source/wiced_ble.c and change the AP_SSID and AP_PASS according to your setup.



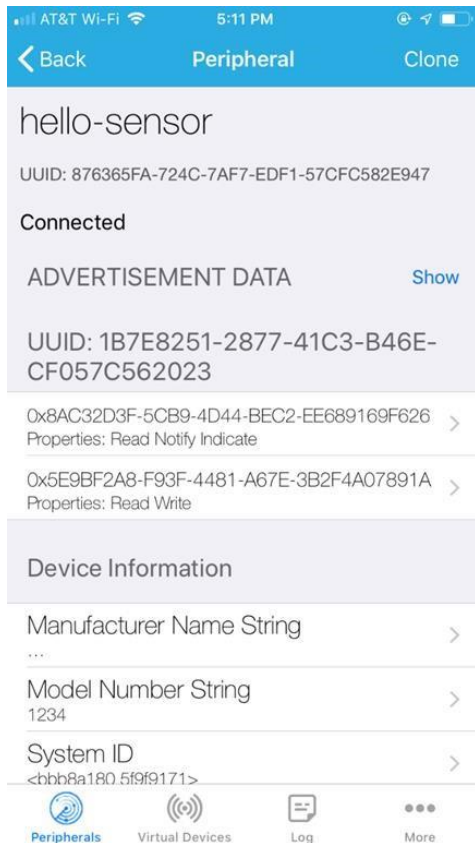
Step 5: Run debug in the IDE. Once the app is booted, BLE advertising starts with the device name "mcuxpresso-hello-sensor". Run Tera Term and connect to the board to see the log.



Step 6: Download and install any BLE scanner apps in your smart phone, such as LightBlue / BLE Scanner / nRF. Open the app and scan for devices. The 'hello-sensor' beacon should show up.



Step 7: Connect to the hello-sensor to view its services. It will start pairing. After the pairing is complete, you will see the following description in the phone app.



Step 8: The Tera Term output will show the following, after the pairing is complete.

```
COM21 - Tera Term VT
File Edit Setup Control Window Help
: TC2_2G
Successfully joined : TC2_2G
Getting IP address from DHCP server
IPv4 Address got from DHCP : 192.168.1.149
Initializing BLE...
Hello Sensor Start Minimum ever free heap size: 34352
hello_sensor_application_init
wiced_bt_gatt_register: 0 wiced_bt_gatt_db_init 0 wiced_bt_cfg_settings.device_name:ncuxpr
esso-hello-sensor wiced_bt_ble_set_advertisement_data 0 wiced_bt_ble_set_scan_response_data 0
3 wiced_bt_start_advertisements 0 Advertisement State Change:
Paired Device Link Keys Request Event Key retrieval success hello_sensor_conn_up id:2
Advertisement State Change: 0 ADU stop Stopping Advertisements0 req_ntu: 185 Paired Device Link Keys Request Event Key retr
ieval success Key Upd Key Upd Key Upd Key Upd Pairing Complete: 0
```

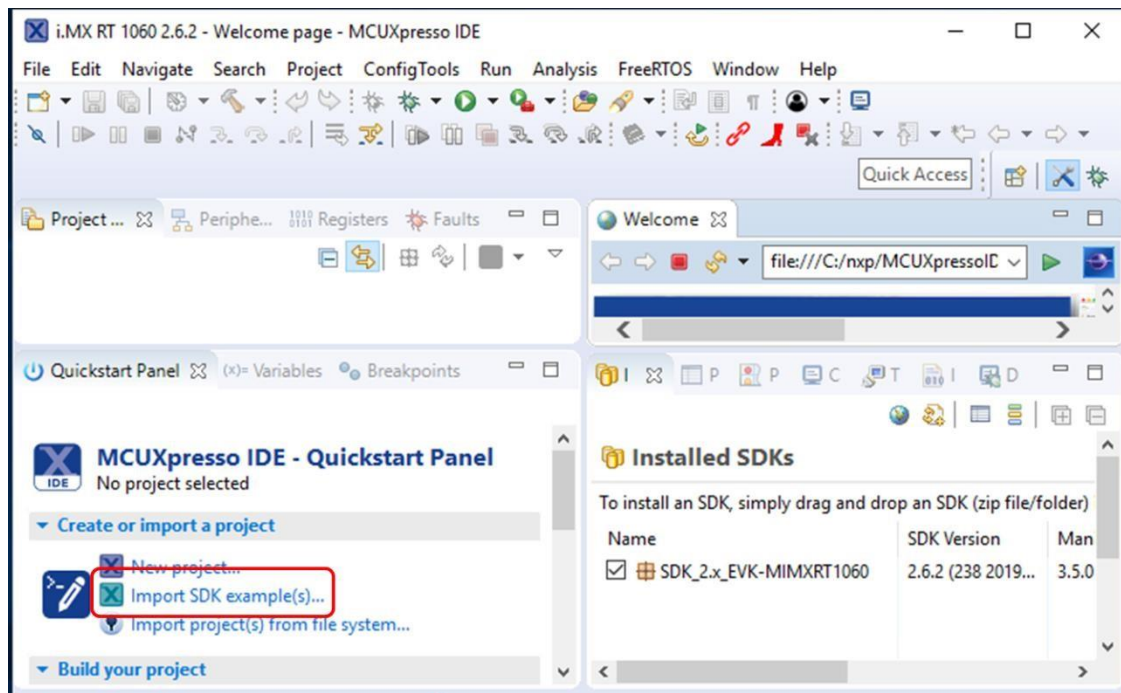
[Back to Table of Content](#)

28. How to run the iPerf sample application on i.MX RT platform?

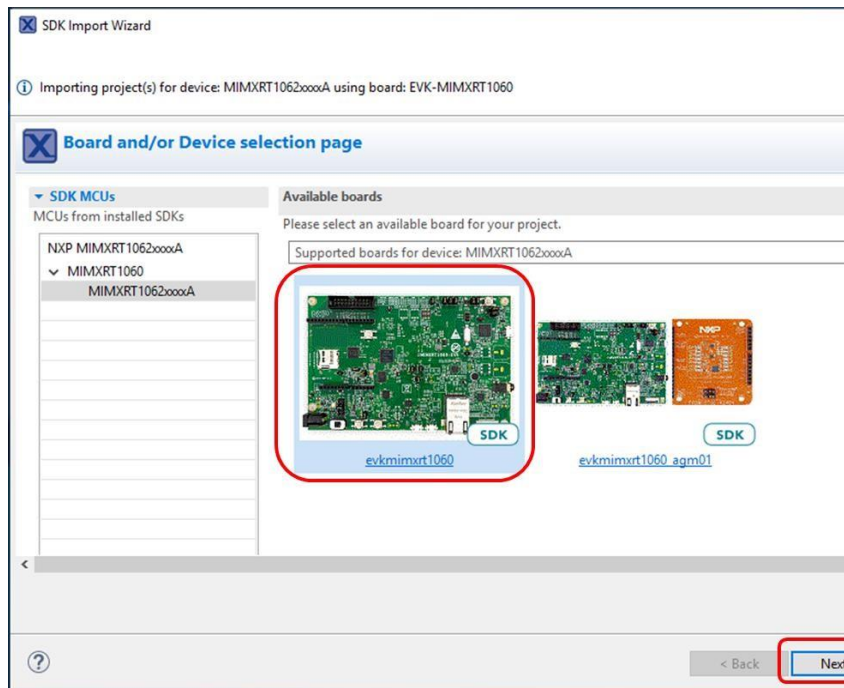
It is assumed you have already set up the MCUXpresso IDE and installed the SDK for your board. You can follow the Murata Quick Start Guide available on [Murata Wireless page](#). The following steps use the RT1060 EVK + 1DX EVB as example.

Step 1: Connect a host PC to the test router via Ethernet cable. This will serve as one of the endpoints of iPerf communication.

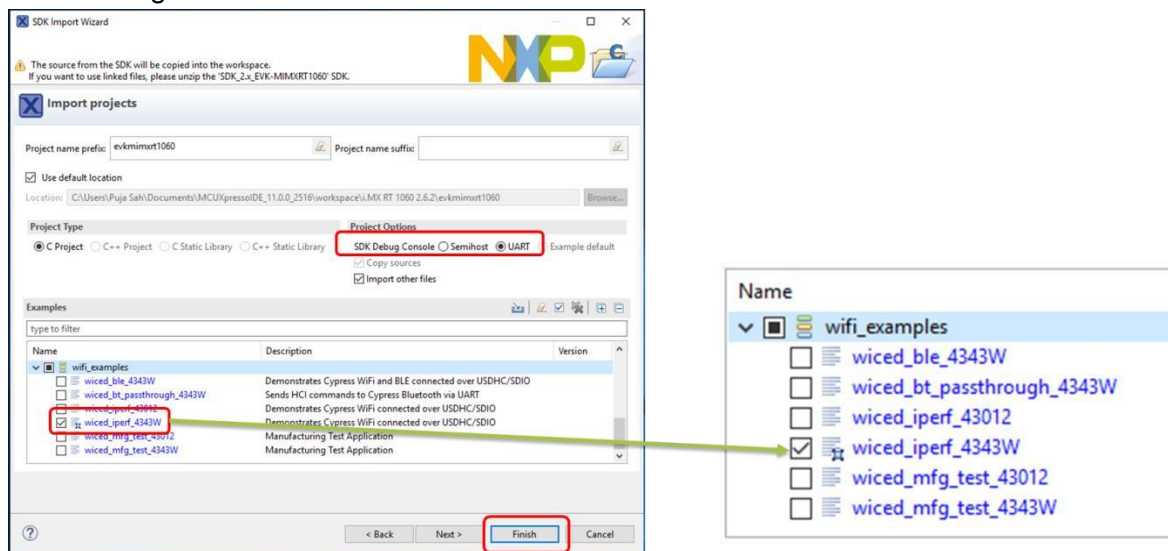
Step 2: Click on “Import SDK example(s)...” in bottom left of IDE inside Quickstart Panel tab.



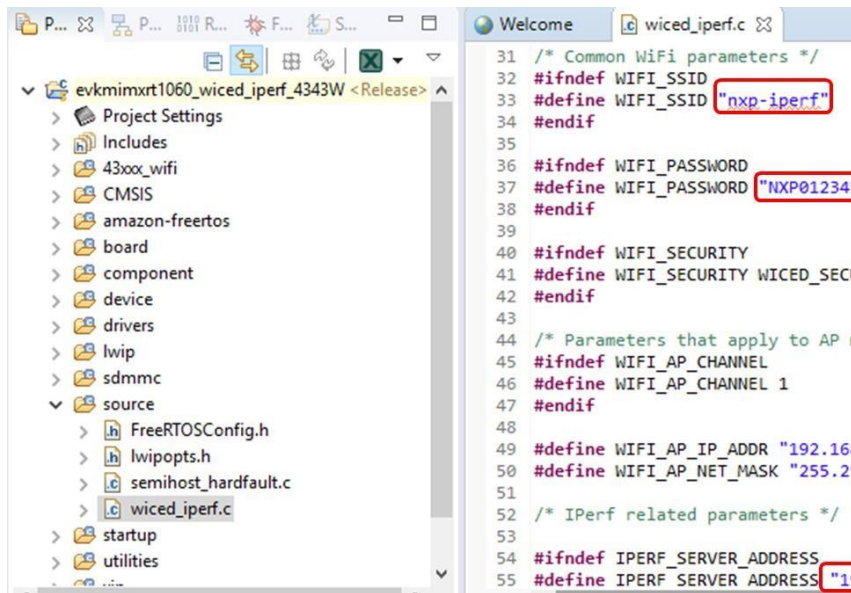
Step 3: Select “evkimxrt1060” board and click “Next” button.



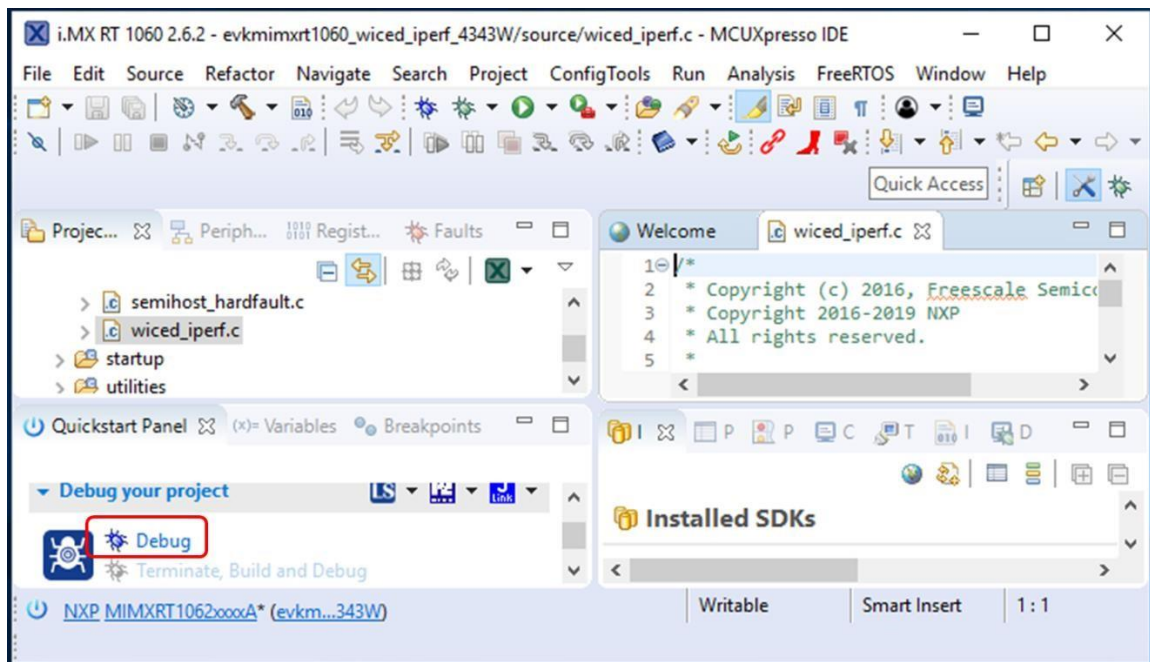
Step 4: Expand wifi_examples. Select wiced_iperf_4343W for 1DX. Select UART for SDK Debug Console. Click Finish button.



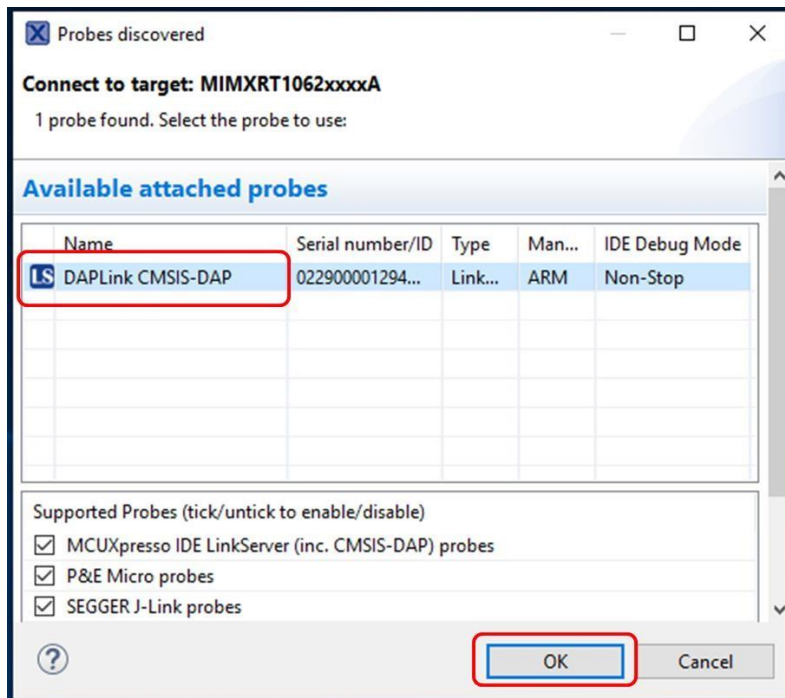
Step 5: Open source/wiced_iperf.c and modify WIFI_SSID, WIFI_PASSWORD and IPERF_SERVER_ADDRESS according to your test router and connected Host PC.



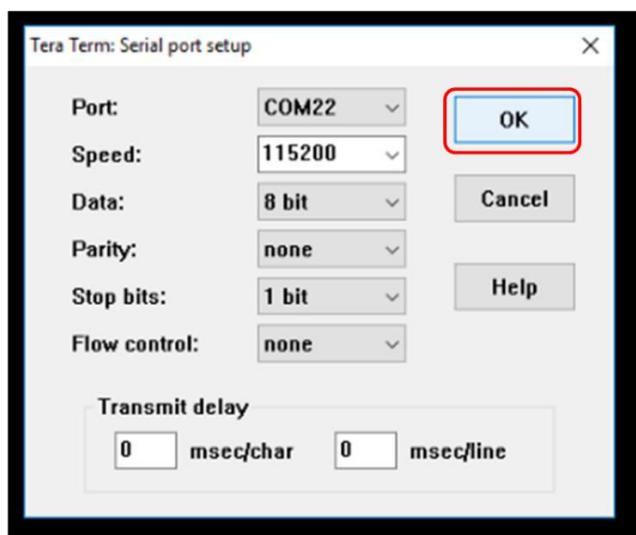
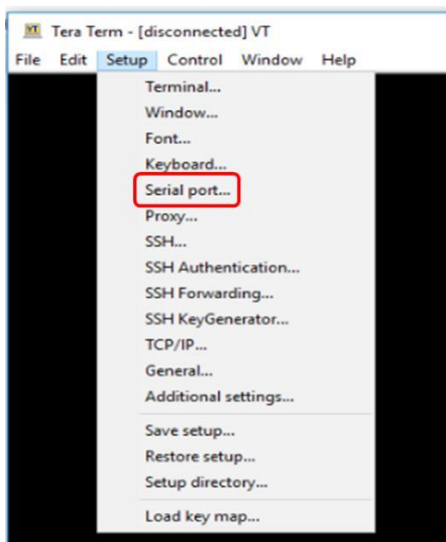
Step 6: Click Debug to run in debug mode.



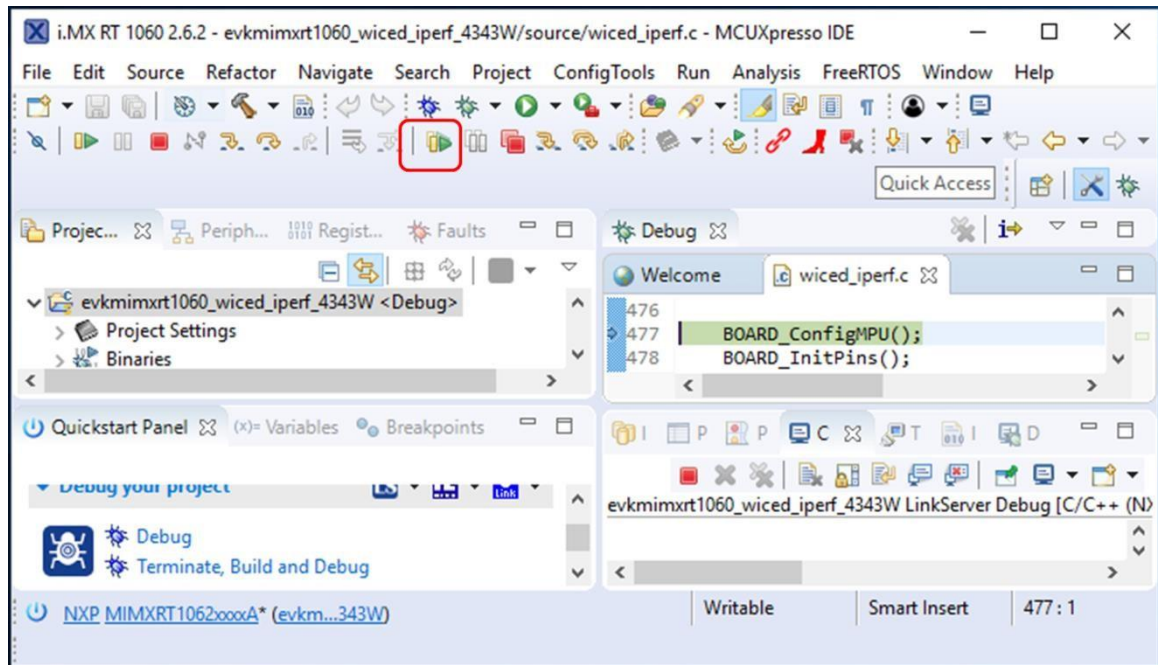
Step 7: Select the appropriate JTAG adapter and hit OK.



Step 8: Run Tera Term or other terminal emulator on appropriate COM port on the Host PC. Set baudrate as 115200 bps, data as 8 bit, parity as none and stop bits as 1 bit.



Step 9: Click on resume debug or hit F8.



Step 10: You will see below output in the terminal window. Type 'c' to run Wi-Fi as client mode.

```
COM27 - Tera Term VT
File Edit Setup Control Window Help
*****
IPERF example
*****
Please select WiFi operation mode:
    a: Access point mode
    c: Client mode
Enter mode:
```

Step 11: To run server mode type “1” in the terminal window.

```

COM21 - Tera Term VT
File Edit Setup Control Window Help

*****
IPERF example
*****

Please select WiFi operation mode:

a: Access point mode
c: Client mode

Enter mode: c
Initializing WiFi connection...

AsyncInterrupt is not supported
WLAN MAC Address : A0:C9:A0:3D:D9:26
WLAN Firmware : wl0: Feb 12 2018 04:08:14 version 7.79.2 (r683798 CV) FWID 01-27b63357
WLAN-CLM : API: 12.2 Data: 9.10.39 Compiler: 1.29.4 ClnImport: 1.36.3 Creation: 20
18-02-12 04:00:50
Successfully initialized WiFi module
Joining: TC2_2G
Successfully joined: TC2_2G
Getting IP address from DHCP server
IPv4 Address got from DHCP : 192.168.1.149

Please select one of the following modes to run IPERF with:

1: TCP server mode (RX only test)
2: TCP client mode (TX only test)
3: TCP client dual mode (TX and RX in parallel)
4: TCP client tradeoff mode (TX and RX sequentially)
5: UDP server mode (RX only test)
6: UDP client mode (TX only test)
7: UDP client dual mode (TX and RX in parallel)
8: UDP client tradeoff mode (TX and RX sequentially)

Enter mode number: 1
Press SPACE to abort the test and return to main menu
  
```

WLAN Firmware

WLAN MAC Address

IP Address

Step 12: Type command “iperf -c <IP address as obtained in previous step> -w 256k -i 1 -P 1” on Host PC, to run the iPerf client. The iPerf test will run.

```

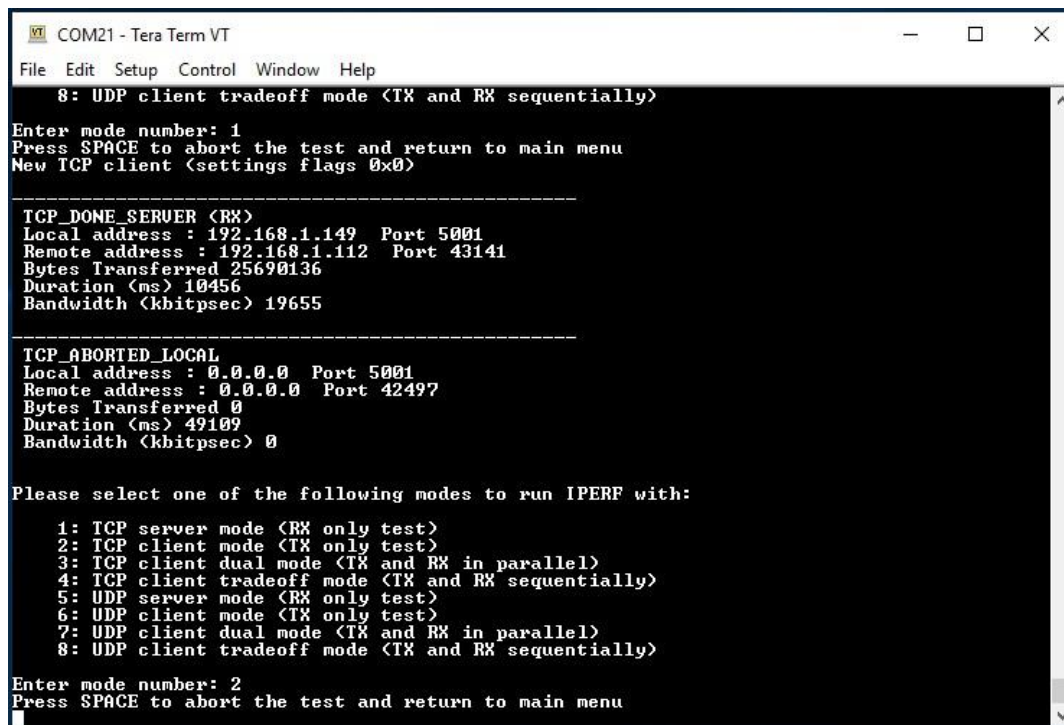
skerr@SDK-W520: ~
skerr@SDK-W520:~$ iperf -c 192.168.1.149 -w 256k -P 1 -i 1
-----
Client connecting to 192.168.1.149, TCP port 5001
TCP window size: 416 KByte (WARNING: requested 256 KByte)
-----
[ 3] local 192.168.1.112 port 43141 connected with 192.168.1.149 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 3] 0.0- 1.0 sec   2.25 MBytes 18.9 Mbits/sec
[ 3] 1.0- 2.0 sec   2.75 MBytes 23.1 Mbits/sec
[ 3] 2.0- 3.0 sec   2.38 MBytes 19.9 Mbits/sec
[ 3] 3.0- 4.0 sec   2.00 MBytes 16.8 Mbits/sec
[ 3] 4.0- 5.0 sec   2.38 MBytes 19.9 Mbits/sec
[ 3] 5.0- 6.0 sec   2.62 MBytes 22.0 Mbits/sec
[ 3] 6.0- 7.0 sec   2.75 MBytes 23.1 Mbits/sec
[ 3] 7.0- 8.0 sec   2.75 MBytes 23.1 Mbits/sec
[ 3] 8.0- 9.0 sec   2.50 MBytes 21.0 Mbits/sec
[ 3] 9.0-10.0 sec   2.00 MBytes 16.8 Mbits/sec
[ 3] 0.0-10.1 sec   24.5 MBytes 20.3 Mbits/sec
skerr@SDK-W520:~$
  
```

Step 13: After iperf completes, you should see below output in console window.

```
Enter mode number: 1
Press SPACE to abort the test and return to main menu
New TCP client <settings flags 0x0>

-----
TCP_DONE_SERVER (RX)
Local address : 192.168.1.149 Port 5001
Remote address : 192.168.1.112 Port 43140
Bytes Transferred 22413336
Duration (ms) 10744
Bandwidth (kbitsec) 16689
```

Step 14: Press Space to return to main menu.



```
COM21 - Tera Term VT
File Edit Setup Control Window Help
8: UDP client tradeoff mode <TX and RX sequentially>

Enter mode number: 1
Press SPACE to abort the test and return to main menu
New TCP client <settings flags 0x0>

-----
TCP_DONE_SERVER (RX)
Local address : 192.168.1.149 Port 5001
Remote address : 192.168.1.112 Port 43141
Bytes Transferred 25690136
Duration (ms) 10456
Bandwidth (kbitsec) 19655

-----
TCP_ABORTED_LOCAL
Local address : 0.0.0.0 Port 5001
Remote address : 0.0.0.0 Port 42497
Bytes Transferred 0
Duration (ms) 49109
Bandwidth (kbitsec) 0

Please select one of the following modes to run IPERF with:
1: TCP server mode <RX only test>
2: TCP client mode <TX only test>
3: TCP client dual mode <TX and RX in parallel>
4: TCP client tradeoff mode <TX and RX sequentially>
5: UDP server mode <RX only test>
6: UDP client mode <TX only test>
7: UDP client dual mode <TX and RX in parallel>
8: UDP client tradeoff mode <TX and RX sequentially>

Enter mode number: 2
Press SPACE to abort the test and return to main menu
```

Step 15: To test the other direction, run Host PC on server mode by typing command "iperf -s -w 256k -i 1".

Step 16: Type 2 in the terminal window. You will see below output in PC.


```
skerr@SDK-W520: ~  
skerr@SDK-W520:~$ iperf -s -w 256k -i 1  
-----  
Server listening on TCP port 5001  
TCP window size: 416 KByte (WARNING: requested 256 KByte)  
-----  
[ 4] local 192.168.1.112 port 5001 connected with 192.168.1.112  
[ ID] Interval      Transfer      Bandwidth  
[ 4] 0.0- 1.0 sec   1.91 MBytes   16.0 Mbits/sec  
[ 4] 1.0- 2.0 sec   1.54 MBytes   12.9 Mbits/sec  
[ 4] 2.0- 3.0 sec   1.79 MBytes   15.0 Mbits/sec  
[ 4] 3.0- 4.0 sec   1.73 MBytes   14.5 Mbits/sec  
[ 4] 4.0- 5.0 sec   1.99 MBytes   16.7 Mbits/sec  
[ 4] 5.0- 6.0 sec   1.88 MBytes   15.8 Mbits/sec  
[ 4] 6.0- 7.0 sec   1.90 MBytes   15.9 Mbits/sec  
[ 4] 7.0- 8.0 sec   1.60 MBytes   13.4 Mbits/sec  
[ 4] 8.0- 9.0 sec   1.98 MBytes   16.6 Mbits/sec  
[ 4] 9.0-10.0 sec   1.89 MBytes   15.8 Mbits/sec  
[ 4] 0.0-10.0 sec   18.2 MBytes   15.2 Mbits/sec
```

[Back to Table of Content](#)

29. How to get better throughput in iPerf on i.MX RT platform?

You can get a better throughput by increasing the TCP window size.

Step 1: Open source/lwipopts.h and locate the lines shown below. Change the PBUF_POOL_SIZE from 9 to 35 (Number of buffers in pool).

```
109 /* ----- Pbuf options ----- */
110 /* PBUF_POOL_SIZE: the number of buffers in the pbuf pool. */
111 #ifndef PBUF_POOL_SIZE
112 #define PBUF_POOL_SIZE 9
113 #endif
```

Step 2: In the same file change the TCP_MSS from (1500 - 40) to 1476 (TCP header size).

```
139 /* TCP Maximum segment size. */
140 #ifndef TCP_MSS
141 #define TCP_MSS (1500 - 40) /* TCP_MSS = (Ethernet MTU - IP header size - TCP header size) */
142 #endif
---
```

Step 3: In the same file change the TCP_WND from 9 to 35 (TCP receive window).

```
155 /* TCP receive window. */
156 #ifndef TCP_WND
157 #define TCP_WND (9 * TCP_MSS)
158 #endif
---
```

Step 4: In the same file, change the DEFAULT_THREAD_PRIO from 8 to 3 (default thread priority).

```
246 * DEFAULT_THREAD_PRIO: The priority assigned to any other lwIP thread.
247 * The priority value itself is platform-dependent, but is passed to
248 * sys_thread_new() when the thread is created.
249 */
250 #ifndef DEFAULT_THREAD_PRIO
251 #define DEFAULT_THREAD_PRIO 8
252 #endif
---
```

Step 5: In the same file, change the TCPIP_THREAD_PRIO from 3 to 8 (TCP thread priority).

```
275 #define TCPIP_MBOX_SIZE 32
276 #define TCPIP_THREAD_STACKSIZE 1024
277 #define TCPIP_THREAD_PRIO 3
278
279 /**
280  * DEFAULT_RAW_RECVMBOX_SIZE: The mailbox size for the :
281  * NETCONN_RAW. The queue size value itself is platform:
282  * to sys_mbox_new() when the recvmbox is created.
```

Step 6: Open source/FreeRTOSConfig.h and locate the lines shown below. Replace the configTOTAL_HEAP_SIZE count from 64 to 35 (total heap size).

```
69 /* Memory allocation related definitions. */
70 #define configSUPPORT_STATIC_ALLOCATION 0
71 #define configSUPPORT_DYNAMIC_ALLOCATION 1
72 #define configTOTAL_HEAP_SIZE ((size_t)(64 * 1024))
73 #define configAPPLICATION_ALLOCATED_HEAP 0
```

Step 7: Save all the changes. Run the IDE in release mode.

Step 8: Follow this FAQ: 7.22, to run the iPerf server test, and you will see the following output on the Host PC running iPerf client.

```
skerr@SDK-W520: ~
skerr@SDK-W520:~$ iperf -c 192.168.1.149 -w 256k -P 1 -i 1
-----
Client connecting to 192.168.1.149, TCP port 5001
TCP window size: 416 KByte (WARNING: requested 256 KByte)
-----
[ 3] local 192.168.1.112 port 43145 connected with 192.168.1.149 port 5001
[ ID] Interval            Transfer          Bandwidth
[ 3] 0.0- 1.0 sec        5.00 MBytes      41.9 Mbits/sec
[ 3] 1.0- 2.0 sec        4.62 MBytes      38.8 Mbits/sec
[ 3] 2.0- 3.0 sec        4.50 MBytes      37.7 Mbits/sec
[ 3] 3.0- 4.0 sec        4.88 MBytes      40.9 Mbits/sec
[ 3] 4.0- 5.0 sec        4.25 MBytes      35.7 Mbits/sec
[ 3] 5.0- 6.0 sec        4.38 MBytes      36.7 Mbits/sec
[ 3] 6.0- 7.0 sec        4.62 MBytes      38.8 Mbits/sec
[ 3] 7.0- 8.0 sec        4.75 MBytes      39.8 Mbits/sec
[ 3] 8.0- 9.0 sec        4.50 MBytes      37.7 Mbits/sec
[ 3] 9.0-10.0 sec        4.75 MBytes      39.8 Mbits/sec
[ 3] 0.0-10.0 sec        46.4 MBytes      38.7 Mbits/sec
skerr@SDK-W520:~$
```

Step 9: If i.MX RT is run as a client, you will see the following output on the Host PC running as iPerf server.

```
skerr@SDK-W520: ~
skerr@SDK-W520:~$ iperf -s -w 256k -i 1
-----
Server listening on TCP port 5001
TCP window size: 416 KByte (WARNING: requested 256 KByte)
-----
[ 4] local 192.168.1.112 port 5001 connected with 192.168.1.112
[ ID] Interval           Transfer             Bandwidth
[ 4] 0.0- 1.0 sec      1.70 MBytes        14.3 Mbits/sec
[ 4] 1.0- 2.0 sec      1.80 MBytes        15.1 Mbits/sec
[ 4] 2.0- 3.0 sec      1.75 MBytes        14.7 Mbits/sec
[ 4] 3.0- 4.0 sec      1.63 MBytes        13.7 Mbits/sec
[ 4] 4.0- 5.0 sec      1.64 MBytes        13.8 Mbits/sec
[ 4] 5.0- 6.0 sec      1.74 MBytes        14.6 Mbits/sec
[ 4] 6.0- 7.0 sec      1.81 MBytes        15.2 Mbits/sec
[ 4] 7.0- 8.0 sec      1.75 MBytes        14.7 Mbits/sec
[ 4] 8.0- 9.0 sec      1.74 MBytes        14.6 Mbits/sec
[ 4] 9.0-10.0 sec      1.86 MBytes        15.6 Mbits/sec
[ 4] 0.0-10.0 sec     17.4 MBytes        14.6 Mbits/sec
```

Same changes can be applied to i.MX RT 1050, i.MX RT 1064 boards for higher throughput numbers.

i.MX RT	1DX				1LV (2G)				1LV (5G)			
	TCP		UDP		TCP		UDP		TCP		UDP	
	Server (Mbps)	Client (Mbps)	Server (Mbps)	Client (Mbps)	Server (Mbps)	Client (Mbps)	Server (Mbps)	Client (Mbps)	Server (Mbps)	Client (Mbps)	Server (Mbps)	Client (Mbps)
1050	39	15	48.2	1.07	39	14	48.1	1.07	51	17	68.8	1.07
1060	40	14	48.8	1.07	40	12	49.1	1.07	51	17	69.0	1.07
1064	39	14	46.9	1.07	39	11	48.9	1.07	47	15	68.9	1.07

[Back to Table of Content](#)

30. What is a CLM Blob file and why do I need it?

The firmware for Cypress Wi-Fi chipsets includes information regarding regulatory constraints. For certain modules this information is kept separately in a binary form known as CLM blob.

CLM stands for “Country Locale Matrix”. It is the Cypress maintained database that represents the regulatory configuration (target power outputs) of Wi-Fi modules with respect to countries, bands, data rates and channels. A CLM blob is an instance of the master CLM database that is specific to a particular product (Wi-Fi module). For some modules (such as CYW43362), the CLM is integrated into the WLAN firmware so it does not require a separate CLM blob file. Otherwise, the CLM Blob file is loaded by the FMAC driver at boot time.

For more details, refer to the [Cypress Wi-Fi CLM Regulatory Manual](#).

[Back to Table of Content](#)

31. I have set the output power target for Channel 1 in the NVRAM file, but it is not taking effect. What can be wrong?

There is an overlap among a few parameters in the CLM (CLM Blob file) and NVRAM configuration files. If a parameter is present in both the files (NVRAM and CLM Blob), the minimum value for that parameter will be considered. For example, if the output power target for Channel 1 is set to 20 in NVRAM but is set to 18 in the CLM Blob file for the product, the value of 18 will be used as the operating condition.

Note: It is highly recommended that customers do not change the default power parameters in NVRAM, as modifications to these values may affect regulatory measurements.

[Back to Table of Content](#)

32. What is the difference between regdb and CLM blob in Linux?

The basic objective of both regdb and CLM Blob are same – providing the kernel / Wi-Fi driver with a regulatory database to use. However, the wireless-regdb mechanism has several disadvantages. For example, every update to the db.txt file that is used to update the regulatory specifications requires kernel recompilation. On the other hand, the CLM Blob file is loaded dynamically by the Cypress FMAC driver on module activation. In this case, the regdb values are ignored and only CLM blob values are considered by the Linux Wi-Fi utilities, such as iw, wpa_supplicant etc.

The kernel configuration CONFIG_CFG80211_INTERNAL_REGDB can be checked to see if the kernel is built with regdb mechanism or not.

[Back to Table of Content](#)

33. How to set up P2P network with Murata modules?

P2P or Wi-Fi Direct can be set up using the wpa_cli tool (part of the wpa_supplicant package). In the steps given below, an i.MX 6 platform is set up as a P2P Group Owner (P2P GO) using a Murata module. Another device (such as another i.MX platform, or a smartphone) is required to act as the P2P Client. Together the P2P GO and Client devices establish a P2P Group.

1. It is a good idea to back up the existing /etc/wpa_supplicant.conf file since it will need to be changed.

```
$ cp /etc/wpa_supplicant.conf /etc/wpa_supplicant.conf.bak
```

2. Make sure WLAN device is configured correctly and the wpa_supplicant is up and running.

3. Invoke the wpa_cli tool to configure the P2P interface:

```
$ wpa_cli -i wlan0
```

4. Remove all network associations:

```
> remove_network all
```

```
OK
```

```
<3>CTRL-EVENT-DISCONNECTED bssid=84:1b:5e:f6:a7:60 reason=3  
locally_generated=1
```

5. Check the status

```
> status
```

```
wpa_state=INACTIVE
```

```
ip_address=192.168.1.3
```

```
p2p_device_address=ba:d7:af:56:61:fc
```

```
address=b8:d7:af:56:61:fc
```

```
uuid=a3178764-c106-57ee-8ec2-6bf2e0607166
```

6. Add the P2P Group:

```
> p2p_group_add
```

```
IPv6: ADDRCONF(NETDEV_UP): p2p-wlan0-0: link is not ready
```

```
OK
```

```
<3>P2P-GROUP-STARTED p2p-wlan0-0IPv6: ADDRCONF(NETDEV_CHANGE):
```

```
p2p-wlan0-0: link becomes ready
```

```
GO ssid="DIRECT-Ih" freq=2437 passphrase="sJjJ4JUR"
```

```
go_dev_addr=ba:d7:af:56:61:fc
```

7. Quit wpa_cli

```
> quit
```

8. The following are now set up:

- a. P2P virtual interface

- b. P2P SSID, with selected channel and secure passphrase needed by another P2P client to associate.

9. Verify the new virtual P2P interface:

```
$ ifconfig
...
p2p-wlan0-0 Link encap:Ethernet HWaddr ba:d7:af:56:e1:fc
inet6 addr: fe80::b8d7:afff:fe56:e1fc/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:23 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:0 (0.0 B) TX bytes:4525 (4.4 KiB)

wlan0      Link encap:Ethernet HWaddr b8:d7:af:56:61:fc
inet addr:192.168.1.3 Bcast:192.168. 1.255 Mask:255.255.255.0
UP BROADCAST MULTICAST MTU:1500 Metric:1
RX packets:1903 errors:0 dropped:0 overruns:0 frame:0
TX packets:769 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:367182 (358.5 KiB) TX bytes:76384 (74.5 KiB)
```

10. Assign address to the P2P interface:

```
$ ifconfig p2p-wlan0-0 192.168.2.1 netmask 255.255.255.0
```

11. Now connect a P2P Client such as smartphone (or similar) and assign an IP address with same subnet address. Either device can now ping each other.

[Back to Table of Content](#)

34. How to check that the meta-murata-wireless layer is successfully added to my Yocto build?

If meta-murata-wireless has been correctly added to your Yocto build, it will show up in the build configuration shown at the start of the build operation:

```
Parsing recipes: 100% |#####|
Parsing of 2428 .bb files complete (0 cached, 2428 parsed). 3269 targets, 218 skipped, 8 masked, 0 errors.
NOTE: Resolving any missing task queue dependencies

Build Configuration:
BB_VERSION      = "1.36.0"
BUILD_SYS       = "x86_64-linux"
NATIVELSBSTRING = "ubuntu-16.04"
TARGET_SYS      = "arm-poky-linux-gnueabi"
MACHINE         = "imx6ulevk"
DISTRO          = "fsl-imx-x11"
DISTRO_VERSION  = "4.9.88-2.0.0"
TUNE_FEATURES   = "arm armv7ve vfp neon callconvention-hard cortexa7"
TARGET_FPU      = "hard"
meta
meta-poky       = "HEAD:0ec241873367e18f5371a3ad9aca1e2801dcd4ee"
meta-oe         = "HEAD:0ec241873367e18f5371a3ad9aca1e2801dcd4ee"
meta-multimedia = "HEAD:dacfa2b1920e285531bec55cd2f08743390aaf57"
meta-freescale  = "HEAD:49ac225a38f6d84519798e3264f2e4d19b84f70a"
meta-freescale-3rdparty = "HEAD:1d6d5961dbf82624b28bb318b4950a64abc31d12"
meta-freescale-distro = "HEAD:0ec6d7e206705702b5b534611754de0787f92b72"
meta-bsp        = "HEAD:0ec6d7e206705702b5b534611754de0787f92b72"
meta-sdk        = "HEAD:d65692ecb3a4136fc1cc137152634e8633ddb3c6"
meta-browser    = "HEAD:d6f9aed41c73b75a97d71bffa060b03a66ee087b1"
meta-gnome      = "HEAD:d6f9aed41c73b75a97d71bffa060b03a66ee087b1"
meta-networking = "HEAD:d6f9aed41c73b75a97d71bffa060b03a66ee087b1"
meta-python     = "HEAD:d6f9aed41c73b75a97d71bffa060b03a66ee087b1"
meta-filesystems = "HEAD:dacfa2b1920e285531bec55cd2f08743390aaf57"
meta-qt5        = "HEAD:32bb7d18a08d1c48873d7ab6332d4cc3815a4dff"
meta-murata-wireless = "HEAD:9f2ea9147cc00db3a02e90e8af0357ea0b481f6e"

Initialising tasks: 100% |#####|
NOTE: Executing SetScene Tasks
NOTE: Executing RunQueue Tasks
WARNING: bzip2-native-1.0.6-r5 do_fetch: Checksum mismatch for local file /home/bchen/NXP/Demo/downloads/bzip2-1.0.6.tar.gz
Cleaning and trying again.
WARNING: bzip2-native-1.0.6-r5 do_fetch: Renaming /home/bchen/NXP/Demo/downloads/bzip2-1.0.6.tar.gz to /home/bchen/NXP/Demo/downloads/bzip2-1.0.6.tar.gz
WARNING: bzip2-native-1.0.6-r5 do_fetch: Checksum failure encountered with download of http://www.bzip.org/1.0.6/bzip2-1.0.6.tar.gz
```

Double check the meta-murata-wireless commit ID (it is the string following the “HEAD:” string) with the latest commit ID in the [Murata github](https://github.com/murata/meta-murata-wireless). Make sure to select the correct branch first. This signifies the latest version of meta-murata-wireless has been added to your Yocto build.

Once built, the following additional software components should be available in the filesystem:

- i. FMAC WLAN firmware: The WLAN firmware should be in **/lib/firmware/brcm/** folder. The naming convention for the firmware is:
“brcmfmac”+<CYW number>+<“-sdio” or “-pcie”>+“.bin”.
(e.g. *brcmfmac43455-sdio.bin* for 1MW)
- ii. CLM blob: The regulatory conformance file should be in **/lib/firmware/brcm/** folder. The naming convention for the CLM blob is:
“brcmfmac”+<CYW number>+<“-sdio” or “-pcie”>+“.clm_blob”.
(e.g. *brcmfmac43455-sdio.clm_blob* For 1MW)

- iii. NVRAM file: Module/chipset configurations and default RF parameters file should be in **/lib/firmware/brcm/** folder. The naming convention for the NVRAM file is:
"brcmfmac"<CYW number><"-sdio" or "-pcie">+".txt".
(e.g. *brcmfmac43455-sdio.txt* for 1MW)
- iv. Bluetooth patchfile: The Bluetooth patchfile should be in **/etc/firmware/** folder. The naming convention for the Bluetooth patchfile is:
"BCM"<Chip ID><Chip version>"+<Module name>+".hcd".
(e.g. *BCM4345C0.1MW.hcd* for 1MW)

[Back to Table of Content](#)

35. How to reach Murata for wireless issues on the Murata modules?

Please send your queries directly to Murata in case it is specific to the module. You can reach Murata by sending mail to wirelessfaq@murata.com.

[Back to Table of Content](#)

36. Where can I find driver documentation for Murata modules?

Murata modules use Cypress FMAC driver for Wi-Fi operation. The FMAC documentation is supplied with the FMAC source code as a README file. Some community maintained documentation is also available online, such as [here](#).

For Bluetooth documentation, you can refer to BlueZ documentations, which can be found in the doc folder of [BlueZ packages](#), or in the Internet (such as [here](#)).

[Back to Table of Content](#)

37. Is there a product selector guide for Murata modules?

Check the table below for comparison of Murata modules against some common criteria.

Criteria	1DX	1FX	1LV	1MW	1CX
I need Wi-Fi	✓	✓	✓	✓	✓
I need Bluetooth	✓	✗	✓	✓	✓
I need 5 GHz WLAN	✗	✗	✓	✓	✓
I need high speed WLAN (802.11n)	✓	✓	✓	✓	✓
I need very high speed WLAN (802.11ac)	✗	✗	✗	✓	✓
I need Bluetooth 5.0	✗	✗	✓	✓	✓
I need low power operation	✓	✓	✓	✓	✓
I need 4K streaming capability (802.11ac)	✗	✗	✗	✓	✓
I need FCC/IC reference certified antenna	✓	✓	✗	✓	✗
I need CE compliance	✗	✗	✓	✓	✗
I need SDIO host interfacing for WLAN	✓	✓	✓	✓	✗
I need PCIe host interfacing for WLAN	✗	✗	✗	✗	✓
I have NXP i.MX 8M/8M Mini EVK	✗	✗	✗	✓	✓
I have NXP i.MX 6 UL/ULL EVK	✓	✓	✓	✓	✗
I have NXP i.MX EVK (except 8M/8M Mini/6UL/6ULL)	✓	✓	✗	✓	✗
I have Embedded Artists i.MX EVK	✓	✓	✓	✓	✓
I have NXP i.MX RT EVK	✓	✗	✓	✗	✗
I have Embedded Artists i.MX RT EVK	✓	✗	✓	✗	✗

[Back to Table of Content](#)

38. Which files do I need to overwrite / rename in the UUU files folder so that UUU can download my custom image?

You will need to copy over the **ea-image-base-imx*.tar.bz2** file from your <build_dir>/tmp/deploy/images/<image_specific_folder>/ to the <uuu location>/files folder if you are planning to use the full_tar.uuu settings. You should also copy over the **imx*.dtb** files if you have made any changes there.

[Back to Table of Content](#)

39. I have followed the Murata Getting Started guide and the image files are created. But the image file has a .bz2 extension? How do I extract it?

Bz2 is a compressed format used to reduce size. It provides better compression than .zip or .gz algorithms, but is slower. The *.sdcard.bz2 file It can be extracted to get the *.sdcard file (which is used by SD card formatters to put the image on an SD card)

If you are using Windows, you can install the [7zip application](#) which can handle .bz2 files.

For Linux / Ubuntu systems, the bunzip2 application can be used:

```
$ bunzip2 xxxx.bz2
```

[Back to Table of Content](#)

40. How to set up the terminal emulator to connect to the i.MX platforms?

Please use the following Serial port settings in your terminal emulator program (e.g. [Tera Term](#)):

- Baud rate: 115200
- Data: 8-bit
- Parity: None
- Stop: 1-bit
- Flow control: None

[Back to Table of Content](#)

41. What are the common Yocto images that can work with Murata modules? What bitbake command should I use to build an image that will work with Murata modules?

In case of Embedded Artists EVKs, the best image to use as of now is the **ea-image-base** image. It includes everything required to test Murata M.2 modules out-of-the-box.

For NXP EVKs, the two most common bitbake builds are given below. You can use any of the following:

- core-image-base
- fsl-image-validation-imx

For Embedded Artists' EVKs, the suggested build is:

- ea-image-base

[Back to Table of Content](#)

42. What are the image files created by a Yocto build? Which one do I need?

Let's take an example. A Yocto build for NXP i.MX 6UL EVK (DISTRO=core-image-base) creates many files in the <build_dir>/tmp/deploy/image/imx6ulevk directory. Among these the important ones are:

- imx6ul*.dtb: These are the DTB files that define board specific configurations.
- core-image-base-imx6ulevk.sdcard.bz2: The compressed version of core-image-base-imx6ulevk.sdcard, which is the SD card image binary that can be flashed on the SD card.
- core-image-base-imx6ulevk.tar.bz2: The rootfs archived.

[Back to Table of Content](#)