Getting Started with M.2 Modules and i.MX RT

Copyright 2020 © Embedded Artists AB

# Getting Started with M.2 Modules and i.MX RT





#### **Embedded Artists AB**

Jörgen Ankersgatan 12 SE-211 45 Malmö Sweden

http://www.EmbeddedArtists.com

#### Copyright 2020 © Embedded Artists AB. All rights reserved.

No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual or otherwise, without the prior written permission of Embedded Artists AB.

#### Disclaimer

Embedded Artists AB makes no representation or warranties with respect to the contents hereof and specifically disclaim any implied warranties or merchantability or fitness for any particular purpose. Information in this publication is subject to change without notice and does not represent a commitment on the part of Embedded Artists AB.

#### Feedback

We appreciate any feedback you may have for improvements on this document. Send your comments by using the contact form: <u>www.embeddedartists.com/contact</u>.

### Trademarks

All brand and product names mentioned herein are trademarks, services marks, registered trademarks, or registered service marks of their respective owners and should be treated as such.

# **Table of Contents**

1 Document Revision History 4
2 Introduction
2.1 Conventions
3 QuickStart Guide
3.1 Step #1: Downloading and Installing MCUXpresso SDK
3.2 Step #2: Mount M.2 Module 11
3.3 Step #3: Example Application - iperf 14
3.3.1 Import wiced-iperf 14
3.3.2 Patch
3.3.3 Configure the Example
3.3.4 Compile and Run/Debug 19
3.3.5 Explanation of the example 20
3.3.6 Troubleshooting
3.4 Step #4: Example Application - Bluetooth
3.4.1 Import wiced_ble_4343W 25
3.4.2 Patch
3.4.3 Configure the Example
3.4.4 Compile and Run/Debug 30
3.4.5 Explanation of the example
3.4.6 Troubleshooting
4 Debug Interface
4.1 J-LINK/J-TRACE Support
4.1.1 Install J-LINK Software
4.1.2 MCUXpresso 10.3.0
4.2 Debug Troubleshooting
5 Improved IPERF performance

# **1** Document Revision History

Revision	Date	Description
PA1	2019-04-30	First release.
PA2	2019-05-03	Updated screen shots and some pictures.
PA3	2019-09-10	Updated screen shots in 3.1. Updated debug troubleshooting in 3.3.6. Added Bluetooth example in section 3.4
PA4	2020-01-14	Updated to match SDK 2.7.0
PA5	2020-01-31	Added note in section 5 about broken instructions

### 2 Introduction

This document describes how to add wireless functionality with M.2 modules to an *iMX RT1062 Developer's Kit.* The kit will be referred to as the *iMX RT Developer's Kit* for the rest of the document.

Additional documentation you need is:

- iMX RT1052/1062 Developer's Kit Program Development Guide called Program
   Development Guide for short in this document
- iMX RT1062 Developer's Kit User's Guide
- iMX RT1052/1062 OEM Board Datasheet
- M.2 Module Datasheet for the specific M.2 module you are using

#### 2.1 Conventions

A number of conventions have been used throughout to help the reader better understand the content of the document.

Constant width text - is used for file system paths and command, utility and tool names.

```
$ This field illustrates user input in a terminal running on the
development workstation, i.e., on the workstation where you edit,
configure and build Linux
```

```
# This field illustrates user input on the target hardware, i.e.,
input given to the terminal attached to the COM Board
```

```
This field is used to illustrate example code or excerpt from a document.
```

This field is used to highlight important information

This chapter is a step-by-step guide to get Wi-Fi up and running in shortest possible time:

- 1. The first step is to download and install the SDK.
- 2. The second step describes how to physically mount the M.2 module.
- 3. The third step describes how to setup, patch and run the example project.

Above are the three simple steps to get up-and-running immediately!

There are a couple of more sections describing different aspects, like debugging and a performance patch for iperf.

#### **iMX RT1052**

Note that as of January 2020 the NXP SDK (2.7.0) includes Wi-Fi support for the 1DX and 1LV M.2 modules and Bluetooth support only for the 1DX M.2 module. Additional Bluetooth support is scheduled for a coming release as well as support for the 1MW and 1LV M.2 modules. Exact release date is TBD.

#### **iMX RT1062**

Note that as of January 2020 the NXP SDK (2.7.0) includes Wi-Fi support for the 1DX and 1LV M.2 modules and Bluetooth support only for the 1DX M.2 module. Additional Bluetooth support is scheduled for a coming release as well as support for the 1MW and 1LV M.2 modules. Exact release date is TBD.

NXP's SDK supports several IDEs but this guide is focusing on the free MCUXpresso IDE from NXP. Combining this document and the more general *Program Development Guide* should make it easy to use another IDE as well.

#### 3.1 Step #1: Downloading and Installing MCUXpresso SDK

This section will walk you through the installation of the MCUXpresso SDK, which is a package of sample software projects (with device drivers and peripheral examples and demos) that will get you started immediately with your i.MX RT software development. Note that even though the name of the SDK suggests the code package is for the MCUXpresso IDE, the sample projects have project files for other IDEs as well, such as Keil uVision/MDK, IAR Embedded Workbench, and more.

First download the MCUXpresso SDK by following this URL: https://mcuxpresso.nxp.com/en/welcome. This is the online SDK builder that makes sure you will get the latest version of the software.

Note that some details in the screen dumps in this section might change over time, but the basic walkthrough steps are the same.

You will need to login to your NXP account. Then click on Select Development Board.



Figure 1 – MCUXpresso SDK Builder

Then select the Boards  $\rightarrow$  i.MX  $\rightarrow$  EVK-MIMXRT1060. This is the NXP evaluation board, but most of the samples will work unmodified on the *iMX RT Developer's Kit*.

Select Board   MCUXpresso SDK	× +					
← → C	← → C  https://mcuxpresso.nxp.com/en/select					
NXP MCUXpresso SI	DK Builder		3 🗭 🗏 🔺 🕹			
SDK Dashboard	Select Development Board Search for your board or kit to get started.					
Select Board	Search by Name					
Q Explore	Search					
ADMINISTRATION  Notifications  Preferences  DOWNLOADS  MCUXpresso IDE  MCUXpresso Config Tools	Select a Device, Board, or Kit  EVK-MCIMX7ULP EVK-MIMX8MM EVK-MIMX8MQ EVK-MIMXRT1015 EVK-MIMXRT1020 EVK-MIMXRT1060 EVK-MIMXRT1064 EVKB-IMXRT1050 MEK-MIMX8QM	(No configuration a	selected)			
	Name your SDK					
Privacy Policy   Terms of U	Jse Contact	© 2019 NXP S	emiconductors. All rights reserved.			

Figure 2 – MCUXpresso SDK Builder - Select Development Board

Select Board   MCUXpresso SDK	• × +					
← → C 🔒 https://mcu	$\leftarrow$ $\rightarrow$ C $\stackrel{\text{a}}{}$ https://mcuxpresso.nxp.com/en/select $\Rightarrow$ O :					
NXP MCUXpresso	SDK Builder			0 🗩 (	■ ▲ ▲	
SDK Dashboard	Select Development Board Search for your board or kit to get started.					
GENERAL						
Select Board	Search by Name	Hard	lware Details			
<b>Q</b> Explore	Search	Boa	ard vice	EVK-MIMXRT10 MIMXRT1062	60	
ADMINISTRATION	Select a Device, Board, or Kit EVK-MIMXRT1015 EVK-MIMXET1020	Cor Dev	re Type / Max Freq vice Memory Size	Cortex-M7F / 60 0 KB Flash 1024 KB RAM	OMHz	
Preferences	EVK-MIMXRT1020 EVK-MIMXRT1050 Deprecated	Ac	tions			
DOWNLOADS	EVK-MIMXRT1060	(	Build MCUXpresso	SDK		
MCUXpresso IDE	EVK-MIMXR11064 EVKB-IMXRT1050		Explore selection	n with Pins tool		
MCUXpresso Config Tools	MEK-MIMX8QM MEK-MIMX8QX	Ć	Explore selection	n with Clocks tool		
	▶ Kits					
	▶ Processors	-				
	Name your SDK					
	SDK_2.5.1_EVK-MIMXRT1060					
	Don't use: <>; *//* in the name of your SDK					
Privacy Policy   Terms c	f Use   Contact		© 2019 NXP Ser	niconductors. All righ	its reserved.	

After selecting the NXP EVK, click on the Build MCUXpresso SDK button.

Figure 3 – MCUXpresso SDK Builder - Build MCUXpresso SDK

Select All Toolchains under the Toolchain/IDE dropdown list. Select SDK version 2.7.0 2019-12-19
--

<b>NXP</b> M	<b>CU</b> Xpresso SD	DK Builde	r					•
🖌 SDK Dash	nboard	<b>SDK</b> Generate	Builder a downloadabl					
GENERAL	ard	Developer Selections he	Environment Se ere will impact files a					
Q Explore		SDK Version	019-12-19	Hardware Details Board Device	EVK-MIN			
ADMINISTRATIC	ns	Search Name, Category, or Description Select All Unselect All					Core Type / Max Freq Device	Cortex-M 0 KB Flat
Preference	es	\$	Name 🗘	Category <sup>▲</sup>	Description	Dependencies	Memory Size	1024 KB
	and IDE		CMSIS DSP Library		CMSIS DSP Software Library		SDK Details SDK Version:	2.7.0 (re
	550 IDE	$\checkmark$	AWS IoT	Middleware	AWS IoT		Host OS:	Windows
Config To	ols		canopen	Middleware	CANopen Stack - MicroCANopen Plus		Middleware:	AWS IoT
MCUXpre Secure Pr	e data sso ovisioning		Crank Storyboard GUI	Middleware	Crank Storyboard GUI Engine		Documentation	FreeMA5 stack
Tool			elQ	Middleware	elQ machine learning SDK containing the ARM CMSIS-NN library (neural network ker (more)		Ø API Reference Manual	

Figure 4 – MCUXpresso SDK Builder - Select Toolchain

Click the Select All button to select all optional software components.

Scroll to the bottom of the page and press the *Download SDK* button. This will give you a complete package with all software.



Figure 5 – MCUXpresso SDK Builder - SDK Builder

Also agree to the Software Terms and Conditions by clicking on the I Agree button.

Download of a (about) 170MByte zip-file will begin.



Figure 6 - MCUXpresso SDK Builder - SDK Downloads

Save the file on your computer. Note the download location. As of January 2020, the file name is **SDK\_2.7.0\_EVK-MIMXRT1060.zip** but the version number (i.e., 2.7.0) will likely increment over time.

The SDK needs to be installed in MCUXpresso before it can be used. To do that start MCUXpresso and then drag-n-drop the SDK archive (**SDK\_2.7.0\_EVK-MIMXRT1060.zip**) on to the "Installed SDKs" tab:

🕅 Installed SDKs 🔀 🔲 Properties	; 🖹 Problems	📃 Console	🖉 Terminal	🚮 Image Info	🙀 Debugger Console	
A Installed SDKs 2 warning	is detected					
To install an SDK, simply drag and drop an SDK (zip file/folder) into the 'Installed SDKs' view. [Common 'mcuxpresso' folder]						
Installed SDKs Available Boards	Available Devices					
Installed SDKs Available Boards Name	Available Devices	DK Version	Ma	nifest Version	Location	

During the installation MCUXpresso will make a copy of the archive that you drag-n-dropped so it is ok to delete the **SDK\_2.7.0\_EVK-MIMXRT1060.zip** afterwards if you don't need it for one of the other IDEs.

#### 3.2 Step #2: Mount M.2 Module

Make sure the *iMX RT Developer's Kit* is **powered off** and then mount the M.2 Module, as illustrated in the picture below:



Figure 1 – M.2 Module on iMX OEM Carrier Board

Do not put too much force/pressure on the M2 screw (into the M.2 connector stand-off) so that the PCB is bent. Bending the PCB too much will damage the board! Use your fingers on the bottom side (while screwing) to give a counter-force, keeping the PCB straight.

The picture below illustrates the typical angle (about 25 degrees) to use when inserting the M.2 module into the connector. The pictures illustrate another carrier board, but the principle is exactly the same, regardless of carrier board.



Figure 2 – M.2 Module on Carrier Board

The picture below illustrates how to use two fingers, placed under the grounding stand-off, to avoid bending the board. Make sure to always use this method to avoid damaging the board.



Figure 3 – M.2 Module on Carrier Board

### 3.3 Step #3: Example Application - iperf

As of January 2020 there are five Wi-Fi example applications in the SDK:

- wiced\_iperf\_43012 this is a performance test application for the 1LV M.2 module
- wiced\_iperf\_4343W this is a performance test application for the 1DX M.2 module
- wiced\_iperf3\_4343W this is a performance test application for the 1DX M.2 module. This
  example is only available in the SDK for the iMX RT1052 and the functionality/configuration is
  the same as for the two other iperf examples (except that it is for the newer iperf3 protocol) so
  it will not be covered by this document
- wiced\_mfg\_test\_43012 this is a special application that is used for RF measurements and it
  will not be covered by this document
- wiced\_mfg\_test\_4343W this is a special application that is used for RF measurements and it will not be covered by this document

Running one of the Wi-Fi examples requires the following steps:

- 1. Import the example application and change flash configuration
- 2. Patch the example code
- 3. Configure the example for your environment (e.g. SSID + password for your network)
- 4. Compile and debug/run

#### 3.3.1 Import wiced-iperf

The following steps will guide you through opening the wiced-iperf application from the SDK.

- 1. Install the SDK as described in section 3.1 if you have not done so already
- 2. Click the "Import SDK example(s)..." link in the Quickstart Panel

🕛 Qui... 🗵 Glo... (x)= Vari... 💁 Bre... 🔚 Out... 🖓 🗄



3. Select the MIMXRT1060 and evkimxrt1060. Click Next to go to the project selector.

Importing project(s) for device: MIMXRT1062xxxA using board: EVK-MIMXRT1060			
Board and/or Device selection page			
Solution So	SDKs for selected MCU Name	SDK Version Manifest	J <sup>a</sup> <sub>2</sub> J <sup>a</sup> <sub>2</sub> ∫ 2
Target Core: cortex-m7	Name	SDK Version Manifest	Ve Location
Description: MIMXRT1062: i.MX® MIMXRT1062 600MHz, 512KB SRAM Microcontrollers	BDK_2.x_EVK-MIMXRT1060	2.5.1 3.4.0	<pre>Common&gt;\SDK_2.5.1_EVK-MIMX</pre>
(MCUs) based on ARM® Cortex®-M7 Core			
0		< Back	lext > Einish Cancel

4. Select the wiced\_iperf example and make sure to switch from Semihost to UART for the SDK Debug Console then click Next to go to the Advanced Settings Page.

SDK Import Wizard	
Import projects	
Project name prefix: evkmimxrt1060	2 Project name suffice
Use <u>d</u> efault location	
Location: F:\temp\2019\2019-09-10_imxrt\ws1\evkmimxrt1060	Browse
Project Type	Project Options
C Project C++ Project C Static Library C++ Static Library	SDK Debug Console  SDK Debug Console  SDK Debug Console  U ART U Copy sources I Import other files
Examples	
type to filter	
Name	Version A
b m = bootloader_examples	
>     demo_apps       >     diver_camples       >     enwin_camples       >     enwin_camples       >     intelfs_examples       > <td>SDIO •</td>	SDIO •
3	< Back Next > Einish Cancel

5. Now on the Advanced Settings page, look at the table for the Memory Configuration:

Туре	Name	Alias	Location	Contraction of the second seco	Drive	
Flash	BOARD_FLASH	Flash	0x6000000	0x800000	MIMXRT1060_SFDP_C	
RAM	SRAM_DTC	RAM	0x2000000	0x20000		
RAM	SRAM_ITC	RAM2	0x0	0x20000		
RAM	SRAM_OC	RAM3	0x20200000	0xc0000		
RAM	BOARD_SDRAM	RAM4	0x8000000	0x2000000		
Add Flash     Add RAM     Split     Join     Delete     Import     Merge     Export						

- Click and change the Size for BOARD\_FLASH to 0x00400000 (both iMX RT1052 and RT1062 have 4MB flash)
- Click in the table cell for the driver and then on the small button that appears. Change the Driver for BOARD\_FLASH to MIMXRT1050-EcoXiP\_ATXP032.cfx using the dropdown menu.



The same flash driver is used for both i.MX RT1052 and RT1062.

 With all the changes made, click ok and then finish to have MCUXpresso complete the project setup

#### 3.3.2 Patch

This section describes how to modify the project to be compatible with the features of the *iMX RT Developer's Kit*, like the EcoXiP.

The SDK supports hyper flash but the *iMX RT Developer's Kit* has EcoXip flash from Adesto. Ideally the EcoXip flash driver should be placed in a new file with \_ecoxip in the name but that would require changes to every project for every IDE in the SDK and that is a huge task. The solution is instead to replace the file content but keep the filename. This way none of the projects have to be modified.

Download the zip-file from the *Resources* tab on the product page on Embedded Artists' website. The name of the file is **imxrt1062** ea files <date>.zip

Unpack the downloaded zip-file in a temporary directory, for example: c:/temp/ea\_files, and then copy the following files from that directory into the project in MCUXpresso. It is typically easiest to do this outside of the IDE using for example Windows Explorer.

File to copy	Destination	Comment
pin_mux.c	board/	Replace existing file
fsl_lpi2c.c	drivers/	New file
fsl_lpi2c.h	drivers/	New file
pca6416.c	source/	New file
pca6416.h	source/	New file
wwd_platform.c	wiced/43xxx_Wi-Fi/WICED/platform/MCU/LPC/WWD/	Replace existing file
wwd_SDIO.c	wiced/43xxx_Wi-Fi/WICED/platform/MCU/LPC/WWD/	Replace existing file

evkmimxrt1060_fle	xip/	Replace existing file
xspi_nor_config.c		

After copying the files into the project the IDE must be told to look for the new files. In MCUXpresso this is done by right clicking the project and selecting *Refresh* in the menu.

#### 3.3.3 Configure the Example

Before the wiced\_iperf example can be used some settings must be changed to allow it to find and connect to your network. These settings are found near the top of the wiced\_iperf.c file and looks like this:



Change the WIFI\_SSID, WIFI\_PASSWORD and WIFI\_SECURITY to match your network.

Change the IPERF\_SERVER\_ADDRESS to the server you want to test against. This option will be explained more in detail later but it is good to know where the setting is in the code.

#### 3.3.4 Compile and Run/Debug

To download and run the application, perform these steps:

1. Click Build in the Quickstart Panel

🕛 Quic	(×)= Glob	(x)= Vari	© <sub>©</sub> Brea…	₽ Outli.	🗖
_					
IDE Pr	Oject: evkmi	mxrt1060_w	Quicksta iced_iperf [D	a <b>rt Pane</b> )ebug]	el .
▼ Create	or import a	project			
	New proje Import SD Import pr	ect YK example(s oject(s) fron	s) n file system		
▼ Build y	our project				
°°	Build Clean	>			
🚽 Debug	your proje	t	0	<li>K - 🔛 ·</li>	- Jink -

- 2. The program builds without errors
- 3. Connect the debug probe (LPC-Link2, ULINK2, J-LINK, etc.) to development platform to your PC via USB cable. See chapter 4 for details how to connect the debug probe.

- 4. Open the terminal application on the PC, such as TeraTerm or PuTTY, and connect to the debug serial port number. Configure the terminal with 115200 baud, 8N1. You can alter the baud rate be searching for the reference BOARD\_DEBUG\_UART\_BAUDRATE variable in file: board.h
- 5. Click the "Debug" button in the Quickstart Panel to download the application to the target.

🕛 Quic	(×)= Glob	(x)= Vari	© <sub>☉</sub> Brea	Be Outli	- [
N IDE Pr	ICUXpres	sso IDE - mxrt1060_w	Quicksta iced_iperf [D	art Panel ebug]	
▼ Create	or import a	project			
	New proje Import SD Import pro	ect )K example(s oject(s) fron	s) n file system.		
▼ Build y	our project				
° <b>°</b>	Build Clean				
🚽 Debug	your projec	t	D	🕻 🕶 🔛 👻	J Link 👻
	F Debug F Terminate	e, Build and	Debug		
▼ Miscel	laneous				

- 6. The application is then downloaded to the target and automatically runs to the main() function.
- 7. Run the code by clicking the "Resume" button to start the application.



8. The wiced\_iperf application is now running and a banner is displayed on the terminal. If this is not true, check your terminal settings and connections. Select Client mode by typing c.

```
M COM49:115200bps - Tera Term - MBED VT
File Edit Setup Control Window Help
TTERF example
TPERF example
TPlease select WiFi operation mode:
    a: Access point mode
    c: Client mode
Enter mode:
```

#### 3.3.5 Explanation of the example

Ping is a great way to test if the hardware is connected to the network, or not, but to really test the network interface it is better to use a program like *iperf*. Iperf requires both a server and a client to run the tests. The client is typically run on the iMX RT1062 board and the server software can either be installed on a computer on the local network (<u>https://iperf.fr/iperf.download.php</u>) or one of the online servers can be used (<u>https://iperf.fr/iperf.servers.php</u>). It is also possible to run the server on the iMX RT1062 and the client on a PC in the same local network.

Make sure to select the latest version of the 2.0.\* branch (2.0.9 as of May 2019) when downloading the iperf software from <u>https://iperf.fr/iperf-download.php</u>. Version 3.\* is not supported by the demo code and it will not work as iperf and iperf3 are completely different protocols.

The wiced\_iperf example implements eight different modes of testing:

```
Successfully joined: VSG1

Getting IP address from DHCP server

IPv4 Address got from DHCP : 192.168.50.212

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)

5: DP client dual mode (TX and RX in parallel)

6: UDP client tradeoff mode (TX and RX sequentially)

5: UDP client tradeoff mode (TX and RX sequentially)

5: UDP client tradeoff mode (TX and RX sequentially)
```

#### Mode 1 - Server RX Only

Type a "1" to start server mode. Check the IP number that was assigned (in this case 192.168.50.164 as shown in the image above) and then run the following command on the PC (the program exists for most common operating systems but is shown here for Windows):

C:\temp> iperf -c 192.168.50.164 -i 2 -t 20 -P 4

The parameters specifies the IP number of the iMX RT1062 board, that a report should be printed every 2 seconds, that the test will run for 20 seconds and use 4 threads/connections.

After the test completes it will print something like this on the PC:

[SUM] 0.0-20.0 sec 92.5 MBytes 37.3 Mbits/sec

The log from the iMX RT1062 will show something like this for each of the 4 connections after the test completes:

```
TCP_DONE_SERVER (RX)
Local address : 192.168.50.164 Port 5001
Remote address : 192.168.50.2 Port 43036
Bytes Transferred 24248344
Duration (ms) 27485
Bandwidth (kbitpsec) 7056
```

The information shown on the target is not entirely correct. It shows the correct number of bytes transferred but does not calculate the time correctly. As seen above the time is 27485ms for a test that takes 20 seconds. The wiced\_iperf code measures time from the connection is established and not from the first byte is sent over the connection. Depending on the PC software it may take extra time (in this case roughly 7.5 seconds) to setup the test and this should not be included in the test result but it is. When checking the result look at the PC instead of the target (i.e., the iMX RT1062).

Press SPACE to return to the main menu to run this or another test again.

#### Mode 2 - Client TX Only

Before running this test the PC must be running the server:

```
C:\temp> iperf -s
Server listening on TCP port 5001
TCP window size: 208 KByte (default)
```

Type a "2" to start client mode. It will run the tests against the server IP number configured in section 3.3.3. Make sure that matches the IP number of the PC running the server.

After the test completes (it takes 10 seconds) it will print the following:

```
TCP_DONE_CLIENT (TX)
Local address : 192.168.50.164 Port 49153
```

```
Remote address : 192.168.50.2 Port 5001
Bytes Transferred 23600924
Duration (ms) 10000
Bandwidth (kbitpsec) 18880
```

Press SPACE to return to the main menu to run this or another test again.

#### Mode 3 - Client RX+TX in Parallel

Before running this test the PC must be running the server:

```
C:\temp> iperf -s
Server listening on TCP port 5001
TCP window size: 208 KByte (default)
```

Type a "3" to start in client mode. It will run the tests against the server IP number configured in section 3.3.3. Make sure that matches the IP number of the PC running the server.

After the test completes (it takes 10 seconds) it will print the following:

```
TCP DONE CLIENT (TX)
Local address : 192.168.50.164 Port 49153
Remote address : 192.168.50.2 Port 5001
Bytes Transferred 16591464
Duration (ms) 10000
Bandwidth (kbitpsec) 13272
TCP_DONE_SERVER (RX)
Local address : 192.168.50.164 Port 5001
Remote address : 192.168.50.2 Port 43052
Bytes Transferred 19004004
Duration (ms) 10520
Bandwidth (kbitpsec) 14448
```

Press SPACE to return to the main menu to run this or another test again.

### Mode 4 - Client RX+TX in Sequence

Before running this test the PC must be running the server:

```
C:\temp> iperf -s
Server listening on TCP port 5001
TCP window size: 208 KByte (default)
```

Type a "4" to start in client mode. It will run the tests against the server IP number configured in section 3.3.3. Make sure that matches the IP number of the PC running the server.

After the test completes (it takes 10 seconds) it will print the following:

```
TCP_DONE_CLIENT (TX)
Local address : 192.168.50.164 Port 49153
Remote address : 192.168.50.2 Port 5001
Bytes Transferred 23441784
Duration (ms) 10000
Bandwidth (kbitpsec) 18752
TCP_DONE_SERVER (RX)
Local address : 192.168.50.164 Port 5001
Remote address : 192.168.50.2 Port 43054
Bytes Transferred 30276044
Duration (ms) 10005
Bandwidth (kbitpsec) 24208
```

Press SPACE to return to the main menu to run this or another test again.

#### Modes 5 to 8 - UDP

The first four modes were for TCP and modes 5 through 8 are for UDP. The tests are run in the same way except for one thing: the -u option must be used on the PC side to force iperf into UDP mode.

```
C:\temp> iperf -s -u
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 208 KByte (default)
```

Or

C:\temp> iperf -u -c 192.168.50.164 -i 2 -t 20 -P 4

### 3.3.6 Troubleshooting

Some things to check if the application is not working as expected:

Program not starting after being flashed.

This is typically because either the flash algorithm/flash size was not changed when importing the project or the project files were not patched.

#### The iperf tests fail.

Check that there in actual connection to the (wireless) network. Look for these prints in the boot log:

```
Joining: VSG1
Successfully joined: VSG1
Getting IP address from DHCP server
IPv4 Address got from DHCP : 192.168.50.212
Please select one of the following modes to run IPERF with:
1: TCP server mode (RX only test)
```

In order for the tests to work the "Successfully joined" message must be shown and the board must have received an IP address from DHCP. If one of those is missing check the SSID information again (see 3.3.3).

#### • Cannot find M.2 module.

A successful detection of the M.2 module looks like this:

```
COM49:115200bps - Tera Term - MBED VT
 <u>File Edit Setup Control Window H</u>elp
 IPERF example
 Please select WiFi operation mode:
    a: Access point mode
    c: Client mode
Enter mode: c
Initializing WiFi connection...
 Assuming the M.2 module is inserted
Setting PCA_WL_REG_ON_1V8 - HIGH
 AsyncInterrupt is not supported
WLAN MAC Address : A0:C9:A0:3D:41:A9

      WLAN Firmware
      : wl0: Feb 12 2018 04:08:14 version 7.79.2 (r683798 CY) FWID 01-

      WLAN CLM
      : API: 12.2 Data: 9.10.39 Compiler: 1.29.4 ClmImport: 1.36.3 Cr

                 : API: 12.2 Data: 9.10.39 Compiler: 1.29.4 ClmImport: 1.36.3 Cre
Successfully initialized WiFi module
```

If the M.2 module cannot be detected, check that the module is correctly inserted and that all patches were applied as specified in section 3.3.2 Make sure that the project matches the M.2 module - the wiced\_iperf\_43012 is for 1LV and wiced\_iperf4343W is for 1DX.

#### • Firewalls

As with all network tests make sure that the firewall on the test PC is not stopping the traffic from the iMX RT1062 board. This is probably not a problem when running the iperf client on the PC as it is outgoing traffic, however when running the iperf server it expects incoming connections and those may be blocked by a firewall.

#### Throughput

The exact speed that can be achieved during testing is heavily dependent on the test environment including but not limited to:

- Use of onboard antenna vs external antenna
- Wireless router performance
- Distance between iMX RT1062 and router
- PC that is running the software
- Usage of the 2.4GHz/5GHz band by others

#### 3.4 Step #4: Example Application - Bluetooth

As of January 2020 there are two Bluetooth example applications in the SDK:

- wiced\_ble\_4343W this is a Bluetooth Low Energy (BLE) + Wi-Fi application for the 1DX M.2 module
- wiced\_bt\_passthrough\_4343W this application allows direct communication over UART between the 1DX M.2 module and a PC. It requires the CyBluetool from Cypress to work and it will not be covered by this document

Running one of the examples requires the following steps:

- 1. Import the example application and change flash configuration
- 2. Patch the example code
- 3. Configure the example for your environment (e.g. SSID + password for your network)
- 4. Compile and debug/run

#### 3.4.1 Import wiced\_ble\_4343W

The following steps will guide you through opening the wiced\_ble\_4343W application from the SDK.

- 1. Install the SDK as described in section 3.1 if you have not done so already
- 2. Click the "Import SDK example(s)..." link in the Quickstart Panel

 U Qui...
 (A) = Glo...
 (A) = Vari...
 Image: Bre...
 Image: Out...
 Image: Out...

3. Select the MIMXRT1060 and evkimxrt1060. Click Next to go to the project selector.

Importing project(s) for device: MIMXRT1062xxxA using board: EVK-MIMXRT1060			
Board and/or Device selection page			
Sold and/or Device selection page  Sold and/or Device selection page  Valiable boards Please select an available board for your project.  Please select an available board for your project.  Please select an available board for your project.  Supported boards MIMART1062000A MIMART106200A MIMART106200A MIMART106200A MIMART106200A MIMART106200A MIMART106200A MIMART106200A MIMART106200A MIMART106200A MIMART1062	SDKs for selected MCU Name	SDK Version Manifest	J <sup>a</sup> <sub>2</sub> J <sup>a</sup> <sub>2</sub> ∫ 2
Target Core: cortex-m7	Name	SDK Version Manifest	Ve Location
Description: MIMXRT1062: i.MX® MIMXRT1062 600MHz, 512KB SRAM Microcontrollers	BDK_2.x_EVK-MIMXRT1060	2.5.1 3.4.0	<pre>Common&gt;\SDK_2.5.1_EVK-MIMX</pre>
(MCUs) based on ARM® Cortex®-M7 Core			
0		< Back	lext > Einish Cancel

4. Select the wiced\_ble\_4343W example and make sure to switch from Semihost to UART for the SDK Debug Console then click Next to go to the Advanced Settings Page.

SDK Import Wizard	
Import projects	
Project name prefix: evkmimxr1060	2 Project name suffix
✓ Use <u>d</u> efault location	
Location: F:\temp\2019\2019-09-10_imxrt\ws2\evkmimxrt1060	Browse
Project Type	Project Options
C Project C++ Project C Static Library C++ Static Library	SDK Debug Console () Semihol () UART V Copy sources V Import other files
Examples	🔤   🖉 🖗 🗎 🖻
type to filter	
Name	Version
Consist driver_examples       C	r USDHC/SDIO
0	< Back Next > Einish Cancel

5. Now on the Advanced Settings page, look at the table for the Memory Configuration:

Туре	Name	Alias	Location	Ciaco Driver	÷
Flash	BOARD_FLASH	Flash	0x6000000	0x800000 MIMXRT1060_SFDP_C	
RAM	SRAM_DTC	RAM	0x20000000	0x20000	
RAM	SRAM_ITC	RAM2	0x0	0x20000	
RAM	SRAM_OC	RAM3	0x20200000	0xc0000	_
RAM	BOARD_SDRAM	RAM4	0x8000000	0x2000000	_
Add Flash Add	RAM	Split Join Delete	j li	mport] Merge] Export] Generate]	

- Click and change the Size for BOARD\_FLASH to 0x00400000 (both iMX RT1052 and RT1062 have 4MB flash)
- Click in the table cell for the driver and then on the small button that appears. Change the Driver for BOARD\_FLASH to MIMXRT1050-EcoXiP\_ATXP032.cfx using the dropdown menu.



The same flash driver is used for both i.MX RT1052 and RT1062.

8. With all the changes made, click ok and then finish to have MCUXpresso complete the project setup

SST39VF3201x\_Hitex\_LPC1850A\_4350A.cfx

#### 3.4.2 Patch

This section describes how to modify the project to be compatible with the features of the *iMX RT* Developer's Kit, like the EcoXiP.

The SDK supports hyper flash but the *iMX RT Developer's Kit* has EcoXip flash from Adesto. Ideally the EcoXip flash driver should be placed in a new file with \_ecoxip in the name but that would require changes to every project for every IDE in the SDK and that is a huge task. The solution is instead to replace the file content but keep the filename. This way none of the projects have to be modified.

Download the zip-file from the Resources tab on the product page on Embedded Artists' website. The name of the file is imxrt1062 ea files <date>.zip

Unpack the downloaded zip-file in a temporary directory, for example: c:/temp/ea files, and then copy the following files from that directory into the project in MCUXpresso. It is typically easiest to do this outside of the IDE using for example Windows Explorer.

File to copy	Destination	Comment
pin_mux_ble.c	board/pin_mux.c	Replace existing file
fsl_lpi2c.c	drivers/	New file
fsl_lpi2c.h	drivers/	New file
pca6416.c	source/	New file
pca6416.h	source/	New file
wwd_platform.c	wiced/43xxx_Wi-Fi/WICED/platform/MCU/LPC/WWD/	Replace existing file
wwd_SDIO.c	wiced/43xxx_Wi-Fi/WICED/platform/MCU/LPC/WWD/	Replace existing file

After copying the files into the project the IDE must be told to look for the new files. In MCUXpresso this is done by right clicking the project and selecting *Refresh* in the menu.

#### 3.4.3 Configure the Example

Before the wiced\_ble\_4343W example can be used some settings must be changed to allow it to find and connect to your network. These settings are found near the top of the wiced\_ble.c file and looks like this:



Change the AP\_SSID, AP\_PASS and AP\_SEC to match your network.

#### 3.4.4 Compile and Run/Debug

To download and run the application, perform these steps:

1. Click Build in the Quickstart Panel

🕛 Quic	(×)= Glob	(x)= Vari	© <sub>☉</sub> Brea	B≊ Outli	
IDE N Pr	CUXprest oject: evkmi	<b>sso IDE -</b> mxrt1060_w	Quicksta iced_iperf [D	a <b>rt Panel</b> Jebug]	l
▼ Create	or import a	project			
	<ul> <li>New proje</li> <li>Import SD</li> <li>Import project</li> </ul>	ect K example(s oject(s) fron	;) n file system		
▪ Build y	our project				
°ô	Build Clean	>			
🚽 Debug	your projec	t	٥	< 🛛 🕶 🔛 🕶	J Link 🔻

- 2. The program builds without errors
- 3. Connect the debug probe (LPC-Link2, ULINK2, J-LINK, etc.) to development platform to your PC via USB cable. See chapter 4 for details how to connect the debug probe.

- Open the terminal application on the PC, such as TeraTerm or PuTTY, and connect to the debug serial port number. Configure the terminal with 115200 baud, 8N1. You can alter the baud rate be searching for the reference BOARD\_DEBUG\_UART\_BAUDRATE variable in file: board.h
- 5. Click the "Debug" button in the Quickstart Panel to download the application to the target.



- 6. The application is then downloaded to the target and automatically runs to the main() function.
- 7. Run the code by clicking the "Resume" button to start the application.



8. The wiced\_ble\_4343W application is now running and some information is displayed on the terminal. If this is not true, check your terminal settings and connections.

```
COM145:115200bps - Tera Term - MBED VT
File Edit Setup Control Window Help
Wi-Fi + BLE example
Initializing WiFi Connection...
 Assuming the M.2 module is inserted
Setting PCA_WL_REG_ON_1V8 - HIGH
 AsyncInterrupt is not supported
WLAN MAC Address : 00:9D:6B:89:EB:76
WLAN Firmware : wl0: Feb 12 2018 04:08:14 version 7.79.2 (r
WLAN CLM
              : API: 12.2 Data: 9.10.39 Compiler: 1.29.4 Cl
Successfully Initialized WiFi Connection
Scanning available networks...
scan completed
                 : Guest-544A9F
#001 SSID
     DCCTD
                  CC.E3.1D.E4.43.30
    Channel
                 : 11
Joining : VSG1
Successfully joined : VSG1
Getting IP address from DHCP server
 IPv4 Address got from DHCP : 192.168.50.83
Initializing BLE...
Hello Sensor Start
Minimum ever free heap size: 34352
Setting PCA BT REG ON 1V8 - HIGH
hello sensor application init
 wiced_bt_gatt_register: 0
wiced bt gatt db init 0
wiced_bt_cfg_settings.device_name:mcuxpresso-hello-sensor
wiced bt ble set advertisement data O
wiced bt ble set scan response data O
Advertisement State Change: 3
wiced bt start advertisements O
```

#### 3.4.5 Explanation of the example

The example will start by scanning for nearby Wi-Fi networks and show some information about each network that is found. It will then connection to the Wi-Fi network specified in section 3.4.3 above.

When the network connection has been established it will start BLE advertising with the device name "mcuxpresso-hello-sensor". Download and install one of the following apps on your phone:

- 1. nRF Connect: https://play.google.com/store/apps/details?id=no.nordicsemi.android.mcp
- 2. LightBlue: https://play.google.com/store/apps/details?id=com.punchthrough.lightblueexplorer

Start the nRF Connect app and press the SCAN button:

		🖇 🕲 🗟 🗐 71% 🛢 13:34
≡ Devid	ces	SCAN :
SCANNER	BONDED	ADVERTISER
No filter		*

The Bluetooth device should appear like this:

🗷 🔟	M	* 🕑 🗟	i <b>l</b> 72% 🛢 1	3:33
	Devices	STOP SC	ANNING	8
sc/	ANNER BOND	ED AD\	/ERTISER	
No fi	lter			•
8	N/A FC:03:9F:E8:65:15 NOT BONDED	▲ -102 dBm	⇔152 ms	
8	mcuxpresso-hell 43:43:A1:12:1F:AC BONDED	o-sensor ⊿ -49 dBm	CONNECT ↔ 14 ms	
0	N/A 24:7B:82:F3:8C:D5 NOT BONDED	<b>⊿</b> -101 dBm	⇔102 ms	
0	N/A 7B:77:6D:26:FE:BB NOT BONDED	<b>⊿</b> -96 dBm	CONNECT ↔ 273 ms	:
Θ	N/A 04:4B:57:3D:5A:44 NOT BONDED	📕 -100 dBm	⇔98 ms	
	N1/A	2		

Click CONNECT to get a list of attributes and information like this:

	1		🕏 🗟 🗟 🕯 🕈	13:33
Ξ	Devices		DISCONNECT	:
ED	ADVERTISER	<b>MCUXI</b> 43:43:A	PREO-SENSOR	×
CONN BOND	NECTED	CLIENT	SERVER	0
Gene UUID: PRIM	eric Attribute 0x1801 ARY SERVICE		_	
Thi	is service is emp	oty.		
Gene	eric Access			
UUID:	0x1800			
PRIM	ARY SERVICE			
De	vice Name		(	+
UU	ID: 0x2A00			
Pro	operties: <b>READ</b>			
Ар	pearance			<u>+</u>
UU	ID: 0x2A01			
Pro	operties: <b>READ</b>			
Unkn	own Service			
UUID:	1b7e8251-287	7-41c3-b46	6e-cf057c562023	6
PRIM	ARY SERVICE			
Devid	ce Information	n		
UUID:	0x180A			

Click the marked button to read the device name. It will look like this:

BONDED		(#);
Generic Attribute		
UUID: 0x1801		
PRIMARY SERVICE		
This service is emp	oty.	
Generic Access		
UUID: 0x1800		
PRIMARY SERVICE		
Device Name		+
UUID: 0x2A00		
Properties: READ		
Value: hello-senso		
Appearance		+
UUID: 0x2A01		
Properties: READ		

(

#### 3.4.6 Troubleshooting

Some things to check if the application is not working as expected:

 Program not starting after being flashed. This is typically because either the flash algorithm/flash size was not changed when importing the project or the project files were not patched.

#### Cannot find M.2 module.

A successful detection of the M.2 module looks like this:

If the M.2 module cannot be detected, check that the module is correctly inserted and that all patches were applied as specified in section 3.3.2 Make sure that the project matches the M.2 module - the wiced\_ble\_4343W is for 1DX.

• Cannot find Bluetooth device when searching with the app.

A successful initialization of the M.2 module looks like this:

```
Initializing BLE...
Hello Sensor Start
Minimum ever free heap size: 34352
Setting PCA_BT_REG_ON_1V8 - HIGH
hello_sensor_application_init
wiced_bt_gatt_register: 0
wiced_bt_gatt_db_init 0
wiced_bt_cfg_settings.device_name:mcuxpresso-hello-sensor
wiced_bt_ble_set_advertisement_data 0
wiced_bt_ble_set_scan_response_data 0
Advertisement State Change: 3
wiced_bt_start_advertisements 0
```

If Bluetooth cannot be initialized, check that the M.2 module is correctly inserted and that all patches were applied as specified in section 3.3.2 Make sure that the project matches the M.2 module - the wiced\_ble\_4343W is for 1DX.

### **4** Debug Interface

It is strongly recommended to use a debug/JTAG probe during program development. The low-cost LPC-Link2 is an excellent choice. Keil ULINK2 and ULINKplus, as well as Segger J-LINK, are also excellent debug probes.

There are two debug interface connectors available on the *iMX OEM Carrier board*:

- J10 this is a Cortex Debug connector. It is a 2x5 pos, 50 mil pitch connector without a shroud. Be careful when inserting the debug probe cable. Position 1 is in specifically marked on the PCB silkscreen. It is located in the lower right corner, see Figure 7 below. The connector supports both the SWD and JTAG interfaces.
- J11 this is an ARM Debug connector. It is a 2x10 pos, 100 mil pitch connector with shroud.

Both connector are defined and supports both the SWD and JTAG type of debug interfaces.

Note that in order to enable the JTAG/SWD interface on the i.MX RT, JP5 shall **not** be shorted/inserted.



Figure 7 - Debug Interfaces on rev B1 iMX OEM Carrier board

Note that due to the powering sequencing requirements on the i.MX RT family, the debug probe I/O voltage **MUST** follow the i.MX RT I/O voltage.

The debug adapter must not drive any output higher than the Vcc/Vref voltage (and if that voltage is zero, then the debug adapter must not drive any output signal). Vcc/Vref is pin 1 on both J10 and J11.

Make sure the debug probe does not have a fixed output voltage, but rather follow Vcc/Vref. If using LPC-Link2 as debug interface, make sure there is **NO** jumper inserted in JP2 on the LPC-Link2.

#### 4.1 J-LINK/J-TRACE Support

This section describes the steps necessary to get the Segger J-TRACE to work with NXP MCUXpresso and Keil uVision. The same instructions are likely to work for Segger J-LINK as well, but it has not been verified.

4.1.1 Install J-LINK Software

The software for the J-TRACE/J-LINK is installed in a central location (i.e. outside of the IDEs). The latest version (v6.60d) as of January 2020 have support for the iMX RT1062 but needs a configuration change to support the EcoXiP flash memory:

- Download and install the latest version of Segger's software: https://www.segger.com/downloads/jlink/JLink\_Windows.exe (v6.42, or later). The rest of the steps will use <jdir> as abbreviation for the folder that the driver was installed in, for example c:\Program Files (x86)\SEGGER\JLink\_V642b\
- 2. Open <jdir>\JLinkDevices.xml
- 3. Search for MCIMXRT1062 to find <Device> entries
- 4. In each of the <Device> entries, change

Loader="Devices/NXP/iMXRT106x/NXP iMXRT106x HyperFlash.elf"

to

Loader="Devices/NXP/iMXRT106x/NXP\_iMXRT106x\_QSPI.elf"

Change the MaxSize="0x04000000" to MaxSize="0x00400000" and then save the file.

MCUXpresso will detect the J-LINK / J-TRACE and configure itself correctly.

If this dialog appears:



Then copy the missing file from the folder that was created in see section 3.3.2 (c:/temp/ea files/)

### 4.2 Debug Troubleshooting

In some cases the IDE complains about not being able to connect to the target. This is most likely because the program already running on the target is interfering. The solution is to put the hardware in ISP mode before starting the flash/debug operation in the IDE. To do this:

- 1. Push and hold down the ISP enable button
- 2. Press the Reset button
- 3. Release the Reset button
- 4. Wait 1 seconds
- 5. Release the ISP enable button

If the LPC-Link2 debugger is used then there are some additional things to note:

- 1. Make sure that the J2 jumper on the LPC-Link2 is not inserted. If the jumper is inserted/closed then the target will be powered by the LPC-Link2 which might be too much power for the usb port that the LPC-Link2 is connected to.
- If the LPC-Link2 is not found by the IDE and you are working on a laptop then try using a powered usb hub instead.
- The troubleshooting section in this forum post has a couple of additional things to try: <u>https://community.nxp.com/thread/388964</u>
- 4. There is a Using and troubleshooting LPC-Link2 in the Appendix Additional Hints and Tips of the User Guide for MCUXpresso IDE. The location of the document is c:\nxp\MCUXpressoIDE\_11.0.0\_2516\MCUXpresso\_IDE\_User\_Guide.pdf if the IDE was installed with the default settings.

## **5** Improved IPERF performance

There are some improvements that can be made to the wiced\_iperf example to more than double the throughput. The changes include rearranging the memory layout, increasing the TCP receive window and increasing the number of buffers.

The instructions below were written for an older version of the SDK. As of 2.7.0 the memory ranges and buffers have been modified and as a result the instructions below will result in a **too big binary that cannot be used**. The instructions are left here as pointers for the reader that wants to experiment with performance tweaking themselves.

- 1) Setup the wiced\_iperf example as explained in 3.3 and verify that it works
- Right click the project and select *Properties…* from the menu. Expand the tree and locate the MCU Settings node:

	MCU settings					<	
Resource							
Builders	Availa	able parts					
C/C++ Build							
Build Variables							
Environment	SDK MCUs			Preinstalle	ed MCIIs		
Logging	MCUs from insta	alled SDKs		MCUs from	n preinstalled LPC ar	nd generic Cortex-M	part suppo
Settings		10524		Terret		-	
Tool Chain Editor		10020004		Target	1102		
C/C++ General	> LPC804				1102		=
MCUXpresso Config Tools	MIMYRT1	150			11 Avv		
Project References	A MIMXRT10	160			11F6x		
Run/Debug Settings	MIMXE	RT1062xxxA			11Exx		
Task Tags				⇒ LPC1	11U6x		
Validation				► LPC	11Uxx		
				→ LPC1	11xx		
				▶ LPC	11xxLV		
				▷ LPC1	1200		
	l arget architectu			contex-m/			
	Preserve men	nory configuration ect configuration : (MIMXRT1062xxxxA)*		cortex-m/			
	Preserve men Preserve proje Memory details Default LinkSe	nory configuration ect configuration (MIMXRT1062xxxxA)* rver Flash Driver		contex-m/			Browse
	Preserve men Preserve proje Memory details Default LinkSe Type	nory configuration ect configuration (MIMXRT1062xxxxA)* rver Flash Driver Name	Alias	Location	Size	Driver	Browse
	Preserve men Preserve proje Memory details Default LinkSe Type Flash	nory configuration ect configuration (MIMXRT1062xxxxA)* rver Flash Driver Name BOARD_FLASH	Alias Flash	Location	Size 0x400000	Driver MIMXRT1050-E.	Browse
	Preserve men Preserve men Preserve proje Memory details Default LinkSe Type Flash RAM	ect configuration ect configuration (MIMXRT1062xxxxA)* rver Flash Driver Name BOARD_FLASH SRAM_DTC	Alias Flash RAM	Location 0x6000000 0x2000000	Size 0x400000 0x20000	Driver MIMXRT1050-E.	Browse
	Preserve men Preserve proji Memory details Default LinkSe Type Flash RAM RAM	ect configuration ect configuration (MIMXRT1062xxxxA)* rver Flash Driver Name BOARD_FLASH SRAM_DTC SRAM_ITC	Alias Flash RAM RAM2	Location 0x6000000 0x0	Size 0x400000 0x20000 0x20000	Driver MIMXRT1050-E.	Browse
	Arget architectuu         Image: Preserve men         Preserve proju         Memory details         Default LinkSe         Type         Flash         RAM         RAM         RAM	inory configuration ect configuration (MIMXRT1062xxxxA)* rver Flash Driver Name BOARD_FLASH SRAM_DTC SRAM_TC SRAM_DC	Alias Flash RAM RAM2 RAM3	Cortex-m/	Size 0x400000 0x20000 0x20000 0xc0000	Driver MIMXRT1050-E.	Browse
	Preserve men Preserve proji Memory details Default LinkSe Type Flash RAM RAM RAM RAM	Inory configuration ect configuration (MIMXRT1062xxxxA)* rver Flash Driver Name BOARD_FLASH SRAM_DTC SRAM_ITC SRAM_ITC BOARD_SDRAM	Alias Flash RAM RAM2 RAM3 RAM4	Cortex-m/	Size           0x400000           0x20000           0x20000           0x20000           0x20000           0x200000	Driver MIMXRT1050-E.	Browse
	arget architectui         Image: Preserve men         Preserve proji         Memory details         Default LinkSe         Type         Flash         RAM         RAM         RAM         RAM         RAM	Inory configuration ect configuration (MIMXRT1062xxxxA)* rver Flash Driver Name BOARD_FLASH SRAM_DTC SRAM_ITC SRAM_ICC BOARD_SDRAM	Alias Flash RAM RAM2 RAM3 RAM4	Cortex-m/	Size           0x400000           0x20000           0x20000           0x20000           0x20000           0x20000           0x20000	Driver MIMXRT1050-E.	Browse
	Arget architectui         Image: Preserve men         Preserve proji         Memory details         Default LinkSe         Type         Flash         RAM         RAM         RAM         RAM         RAM         RAM         RAM	Add RAM	Alias Flash RAM RAM2 RAM3 RAM4 Plit Join Delet	Cortex-m/ Location 0x6000000 0x2000000 0x0 0x2020000 0x8000000 te Imp	Size           0x400000           0x20000           0x20000           0x20000           0x20000           0x200000           0x200000           ort           Merge	Driver MIMXRT1050-E.	Browse
	Arget architectur         Preserve men         Preserve proji         Memory details         Default LinkSe         Type         Flash         RAM         RAM	Add RAM	Alias Flash RAM RAM2 RAM3 RAM4 Plit Join Delet	Location 0x6000000 0x0 0x20200000 0x80000000 0x80000000 0x80000000	Size 0x400000 0x20000 0x20000 0x200000 0x2000000 0x2000000 0x2000000	Driver MIMXRT1050-E. xport) Generate Refresh	Browse
	Arget architectui         Preserve men         Preserve proji         Memory details         Default LinkSe         Type         Flash         RAM	Add RAM	Alias Flash RAM RAM2 RAM3 RAM4 Plit Join Delet	Location 0x6000000 0x2000000 0x0 0x2020000 0x8000000 te Impr	Size 0x400000 0x20000 0x20000 0x200000 0x2000000 0x2000000	Driver MIMXRT1050-E. Aport) Generate Refresh	Browse

Delete the SRAM\_ITC and then change the sizes of SRAM\_DTC and SRAM\_OC so that it

looks like below and then save the changes and close the dialog

Туре	Name	Alias	Location	Size	Driver
Flash	BOARD_FLASH	Flash	0x60000000	0x400000	MIMXRT1050-E
RAM	SRAM_DTC	RAM	0x20000000	0x80000	
RAM	SRAM_OC	RAM2	0x20200000	0x80000	
RAM	BOARD_SDRAM	RAM3	0x80000000	0x2000000	

Open board/board.c and locate these lines that control the size of the SRAM\_DTC (region 5):

577	
378	/* Region 5 setting: Memory with Normal type, not <u>shareable</u> , outer/inner write back */
379	MPU->RBAR = ARM_MPU_RBAR(5, 0x20000000U);
380	MPU->RASR = ARM_MPU_RASR(0, ARM_MPU_AP_FULL, 0, 0, 1, 1, 0, ARM_MPU_REGION_SIZE_128KB);
201	

Replace the ARM\_MPU\_REGION\_SIZE\_128KB to ARM\_MPU\_REGION\_SIZE\_512KB. This aligns the size with the 0x80000 value entered in the previous dialog for SRAM\_DTC.

4) Open source/lwipopts.h and locate these lines:

```
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
```

Change from 9 to 44 buffers in the pool. This is what the increase of SRAM\_DTC was used for.

5) In the same file (source/lwipopts.h) locate these lines:

```
155 /* TCP receive window. */
156 #ifndef TCP_WND
157 #define TCP_WND (9 * TCP_MSS)
158 #endif
159
```

Replace the constant 9 with 44

6) Open startup/startup\_mimxrt1062.c and locate these lines:

```
424
     _attribute_ ((used, section(".isr_vector")))
425 void (* const g_pfnVectors[])(void) = {
426 // Core Level - CM7
427
      &_vStackTop,
                                          // The initial stack pointer
428
      ResetISR,
                                          // The reset handler
429
        NMI_Handler,
                                          // The NMI handler
430
        HardFault Handler,
                                          // The hard fault handler
431
        0,
                                          // Reserved
                                          // Deconved
432
        ø
```

Replace the &\_vStackTop with (0x20200000 + 0x1000U) to move the initial stack pointer to the end of the first block of SRAM\_OC. It should look like this after the change:

```
__attribute__ ((used, section(".isr_vector")))
424
425 void (* const g_pfnVectors[])(void) = {
      // Core Level - CM7
426
427
        (0x20200000 + 0x1000U),
                                          // The initial stack pointer
428
        ResetISR,
                                           // The reset handler
429
        NMI_Handler,
                                            // The NMI handler
430
        HardFault_Handler,
                                           // The hard fault handler
431
                                            // Reserved
        0,
420
        0
                                            // Deconved
```

7) In the same file (startup/startup\_mimxrt1062.c) locate the ResetISR function:

```
646⊖ __attribute__ ((section(".after_vectors.reset")))
647 void ResetISR(void) {
648
649 // Disable interrupts
650 __asm volatile ("cpsid i");
651
652 #if defined (__USE_CMSTS)
```

Insert the following lines after the \_asm... line:

```
*((volatile unsigned int *)0x400ac044) = 0x5aaaaaaa;
*((volatile unsigned int *)0x400ac040) = 0x00200006; // DTC RAM with flexram config
*((volatile unsigned int *)0x400ac038) = 0x00A00000; // DTC - 512, ITC - 0
```

so that it now looks like this:

```
attribute
                 ((section(".after_vectors.reset")))
646
647 void ResetISR(void) {
648
649
       // Disable interrupts
650
        __asm volatile ("cpsid i");
651
        *((volatile unsigned int *)0x400ac044) = 0x5aaaaaaa;
652
       *((volatile unsigned int *)0x400ac040) = 0x00200006; // DTC RAM with flexram config
653
        *((volatile unsigned int *)0x400ac038) = 0x00A00000; // DTC - 512, ITC - 0
654
655
and Hif doffmod / HEE CMETEL
```

8) Compile and run. Compare iperf throughput results with, and without, this patch.