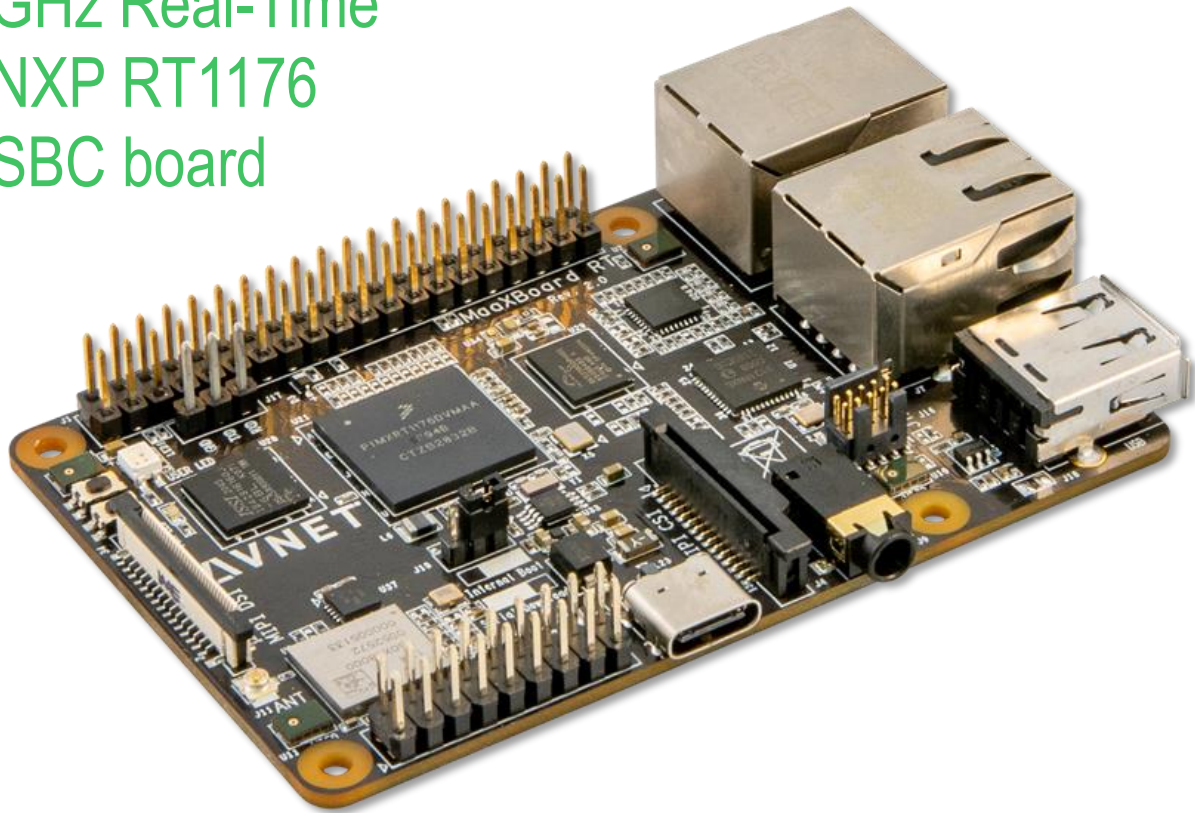


GHz Real-Time
NXP RT1176
SBC board



MaaXBoard RT Hardware User Guide

v1.0 – Sept 21, 2021

1 Document Control

Document Version: v1.0
Document Date: 09/21/2021
Document Author: Peter Fenn
Document Classification: Public
Document Distribution: Public

2 Version History

Version	Date	Comment
1.0	09/21/2021	Release version (for Rev.3 production PCB)

Contents

1	Document Control.....	2
2	Version History	2
3	Hardware Checklist	7
4	Software Checklist.....	7
5	Introduction.....	8
5.1	MaaXBoard RT Info	9
5.2	Items Included with MaaXBoard RT	9
5.3	Important Reference Documents	9
5.4	MCU-LINK Debugger/Programmer Probe (<i>Required Option</i>)	9
5.5	MCU-LINK-PRO Debugger/Programmer Probe (<i>Optional</i>)	10
5.6	MIPI DSI 7-inch Capacitive Touch LCD Display (<i>Optional</i>).....	11
5.7	MIPI CSI 5 MP Camera (<i>Optional</i>)	11
5.8	Other SBCs and SOMs from Avnet Boards (<i>Optional</i>).....	12
5.9	Hardware Setup for Application Development	13
6	MaaXBoard RT Architecture & Features	14
6.1	Features.....	14
6.2	Block Diagram – NXP RT1176 Processor.....	15
6.3	Block Diagram – MaaXBoard RT (<i>Rev.3 PCB</i>)	16
6.4	Location of Components on MaaXBoard RT.....	17
6.5	Jumpers, LEDs, and Switches	18
6.5.1	Boot Mode Configuration Jumper (J19).....	18
6.5.2	Button Switches	18
6.5.3	Status LEDs	18
6.6	Memory Resources	19
6.6.1	Memory Maps for Application Development.....	20
6.6.2	Internal ROM/RAM memory map	20
6.6.3	HyperFlash Memory.....	21
6.6.4	HyperFlash Programming using NXP MCUBootUtility Application.....	23
6.6.5	HyperFlash Programming using MCUXpresso Secure Provisioning Tools.....	25
6.6.6	On-chip TCM RAM and OCRAM	26
6.6.7	SDRAM	26
6.7	Wireless Connectivity.....	27
6.7.1	Wi-Fi SDIO Interface.....	30

6.7.2	BT/BLE UART Interface	30
6.7.3	BT PCM Audio Interface	30
6.7.4	Wi-Fi / BT antenna	30
6.8	Peripheral Devices and Interfaces	31
6.8.1	USB 2.0 Host Interface	31
6.8.2	USB 2.0 Device Interface.....	31
6.8.3	GbE Ethernet (with TSN time-sync.).....	31
6.8.4	10/100 Ethernet (with IEEE 1588 time-sync.).....	31
6.8.5	CAN-FD.....	31
6.8.6	MIPI-DSI Display and Touchscreen.....	31
6.8.7	MIPI-CSI Camera.....	32
6.8.8	Digital Microphones	32
6.8.9	Audio Codec	32
6.8.10	J9: Stereo Audio Jack.....	32
6.8.11	J1: Pi-HAT compatible 40-pin header.....	33
6.8.12	Pi HAT Expansion Boards	33
6.8.13	MikroE Click Boards.....	33
6.8.14	J15: Custom 18-pin expansion header	34
6.8.15	J17: UART1 USB Serial Port 3-pin header.....	35
6.8.16	SWD/JTAG debugger 10-pin mini-header	35
6.9	Power Input, Protection and Regulation.....	36
6.9.1	USB type-C Connector	36
6.9.2	ESD Protection	36
6.9.3	Power Regulation.....	36
6.9.4	Measuring Power Consumption.....	36
7	Development Software Installation	37
7.1	NXP MCUXpresso IDE	37
7.2	NXP MCUXpresso SDK	37
7.3	NXP GUI Guider.....	37
7.4	NXP MCUBootUtility	37
8	Development Environment	38
8.1	Importing a Project zip File.....	38
8.2	Building projects.....	39
8.3	Setting up the debugger.....	39
8.4	Downloading and running the application	40
9	Porting NXP RT1170-EVK SDK examples to MaaXBoard RT	41

9.1	Building and running SDK RT1170-EVK Examples	42
10	MaaXBoard RT Example Applications.....	43
10.1	Custom System-level Reference Designs.....	43
10.2	Adaptations of NXP RT1170-EVK SDK Examples.....	43
11	Customizing the OOB Test Suite (Ref. Design #1).....	44
11.1	Overview of MaaXBoard-RT Reference Design #1.....	44
11.2	Console mode (CLI / Command-Line Interface).....	46
11.3	GUI mode (Graphical UI)	47
11.4	Project Structure for Reference Design #1.....	48
11.5	Adding a new CLI Command	50
12	Customizing the Wi-Fi Webserver (Ref. Design #2)	51
12.1	Overview of the MaaXBoard-RT Reference Design #2.....	51
12.2	Modes of Operation.....	52
12.2.1	HyperFlash Partitioning.....	52
12.2.2	Soft AP mode.....	53
12.2.3	Wi-Fi Client Mode	53
12.3	Project Structure for Reference Design #2.....	54
12.3.1	M7 (MASTER) Wi-Fi Webserver project.....	55
12.3.2	M4 (SLAVE) Sensor Service project.....	55
12.4	Installing the Click Boards	56
12.5	Running the demo	56
12.6	HyperFlash as runtime storage	57
12.7	User Customization.....	58
12.7.1	Frontend (Webserver UI)	58
12.7.2	Backend (LWIP, CGI, HTTP).....	58
12.7.3	Webserver access via Smartphone	59
13	Known Issues	60
14	Cautionary Notes.....	60
15	Technical Support.....	61
15.1	NXP-hosted Technical Support Resources.....	61
15.2	Avnet-hosted Technical Support Resources	61
16	Sales Contact Info	61
17	Disclaimer.....	62
18	Safety Warnings	62

Figures

Figure 1 – NXP MCU-LINK Debug/Programmer Probe	9
Figure 2 – NXP MCU-LINK-PRO Debug/Programmer Probe	10
Figure 3 – Typical hardware setup for software development.....	13
Figure 4 – NXP RT1176 Processor Block Diagram	15
Figure 5 – MaaXBoard RT Block Diagram.....	16
Figure 6 – LEDs and Button Switches	18
Figure 7 – RT1176 on-chip memory resources.....	19
Figure 8 – NXP MCU Boot Utility Settings for Erase of Flashloader	24
Figure 9 – Mass Erase / Reload of Default Flash Driver	25
Figure 10 – Murata Type-1ZM Wi-Fi/BT Combo Module Block Diagram	27
Figure 11 – Schematic Detail of Wi-Fi/BT Implementation.....	29
Figure 12 – Current-measurement USB Dongle	36
Figure 13 – Configure GUI_EN in global.h.....	46
Figure 14 – Default list of supported CLI Menu commands	46
Figure 15 – Screen orientation in project settings	47
Figure 16 – Project folders.....	49
Figure 17 – HyperFlash Base address.....	52

Tables

Table 1 – Hardware Checklist.....	7
Table 2 – Software Checklist	7
Table 3 – Location of Key Components on MaaXBoard RT	17
Table 4 – Memory map settings when using RAM and SDRAM.....	20
Table 5 – Wi-Fi/BT Module Interface (mapped to SDK/EVK signal names).....	28
Table 6 – BT PCM Audio Interface (mapped to SDK/EVK signal names).....	30
Table 7 – Pi-HAT compatible 40-pin header (J1).....	33
Table 8 – Custom 18-pin Expansion header (J15).....	34
Table 9 – UART1 console debug VCOM 3-pin header (J17)	35
Table 10 – SWD/JTAG debugger 10-pin mini-header (J16)	35

3 Hardware Checklist

Hardware items recommended for application development are the following

#	Item Description
1	Computer (Windows / Linux / Mac) with installed development tools (see below)
2	Avnet MaaXBoard RT board http://avnet.me/MaaXBoard-RT
3	NXP MCU-LINK Debugger/Programmer Probe (plus USB type-A to MicroUSB cable)
4	5V 1A power adapter (plus USB type-A to USB type-C cable)
5	USB current monitor dongle (<i>optional</i>)
6	MaaXBoard MIPI-DSI 7" Capacitive Touch 720 x 1280 Display (<i>optional</i>) p/n: AES-ACC-MAAX-DISP1
7	MaaXBoard MIPI-CSI Camera, 5 Mpixel, OV5640 image sensor (<i>optional</i>) p/n: AES-ACC-MAAX-CAM1

Table 1 – Hardware Checklist

4 Software Checklist

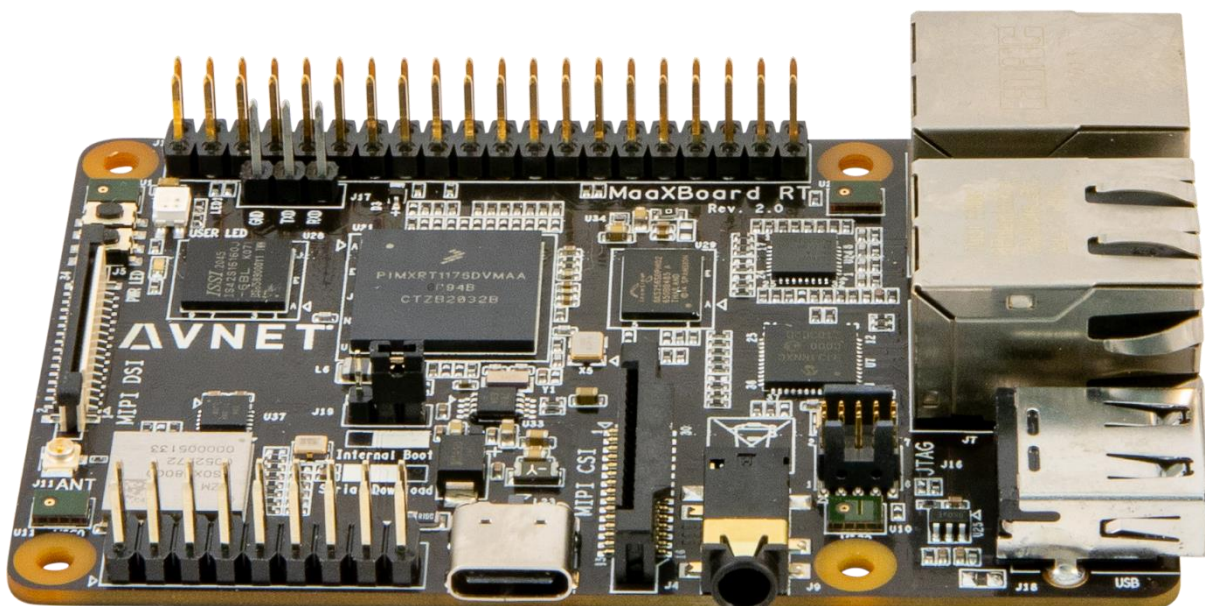
Listed below are the software items mentioned in this document

#	Item Description
1	NXP MCUXpresso IDE (MCUXpressoIDE_11.4.1 or later) https://mcuxpresso.nxp.com
2	NXP MCUXpresso SDK (SDK_2.10.1_MIMXRT1170-EVK or later) Search by name: “ RT1170-EVK ” on NXP SDK builder site at: https://mcuxpresso.nxp.com/en/select Select MIMXRT1170-EVK then click Build MCUXpresso SDK_2.10.1 to download
3	NXP GUI Guider (v1.2.0 or later), LVGL Editor & Code Generation tool) https://www.nxp.com/GUI-GUIDER
4	NXP MCUBootUtility (v3.3.1 or later) https://github.com/JayHeng/NXP-MCUBootUtility
5	Replace files of the same names in NXP SDK examples. Download the files from: http://avnet.me/MaaXBoard-RT-SDK xip folder: (<i>changes to target Hyperflash memory</i>) MaaXBoard_S26KS256.cfx (<i>replaces MIMXRT1170_SFDP_QSPI.cfx</i>) evkmimxrt1170_flexspi_nor_config.c board folder: board.h (<i>changed to define default MaaXBoard RT interfaces</i>) dcd.c (<i>changed to define x16 width SDRAM interface</i>)

Table 2 – Software Checklist

5 Introduction

- **1GHz SBC with advanced real-time capabilities**
 - Use as a processing sub-assembly in OEM products or as a versatile development board
 - Raspberry Pi-4B form-factor facilitates easier enclosure and cabling integration
 - 40-pin HAT compatible expansion header (ie. access Pi HAT ecosystem add-on boards)
 - 18-pin custom expansion connector
- **Based on NXP i.MX RT1176 Crossover MCU**
 - Cortex-M7 @1GHz and Cortex-M4 @400 MHz
- **Board is well resourced with high-speed memories**
 - On-chip: 2 MB RAM (TCM and OCRAM)
 - Onboard: 32 MB SDRAM
 - Onboard: 32 MB HyperFlash
- **Comprehensive set of board interfaces and peripheral devices**
 - 40-pin Pi-HAT compatible expansion connector
 - 18-pin custom expansion connector
 - 1x USB Host interface (type-A connector)
 - 1x USB Device interface (type-C connector)
 - 1x GbE Ethernet (with TSN)
 - 1x 10/100 Ethernet (with IEEE1588)
 - SWD/JTAG debugger 10-pin mini header (use NXP MCU-Link, or 3rd party debug probe)
 - MIPI-DSI touch display interface (2-lane, supports up to 1280x800 display)
 - MIPI-CSI camera interface (2-lane, pinout is same as on Raspberry-Pi)
 - Audio subsystem with 4x onboard digital microphones plus stereo audio output jack
 - 802.11ac Wi-Fi and Bluetooth 5
 - U.FL connected external antenna
 - High efficiency 5V to 3.3V dc/dc buck convertor, plus low-current low-voltage LDOs
 - Operating Temperature: -30~85°C
 - Dimensions: 85mm x 56mm (same as Raspberry-Pi 4)



5.1 MaaXBoard RT Info

- Part# to order: AES-MC-SBC-IMXRT1176-G
- Product Page: <http://avnet.me/MaaXBoard-RT>

5.2 Items Included with MaaXBoard RT

- MaaXBoard RT board
- QuickStart Card
- Downloadable examples, reference designs and documentation

5.3 Important Reference Documents

- MaaXBoard RT QuickStart Card
- MaaXBoard RT Product Brief
- MaaXBoard RT Hardware User Guide (*this document*)
- MaaXBoard RT Schematic and BOM (*available under NDA*)
- NXP IMXRT1170RM

5.4 MCU-LINK Debugger/Programmer Probe (Required Option)

The [NXP MCU-LINK Debugger Probe](#) configured for CMSIS-DAP protocol, is supported by multiple IDEs and is available for purchase separately from Avnet

Standard MCU-Link features:

- High speed USB,
- SWD debug,
- SWO profiling,
- VCOM (USB to UART bridge)

Part# (and link): [MCU-LINK](#) (MSRP = \$10.99)

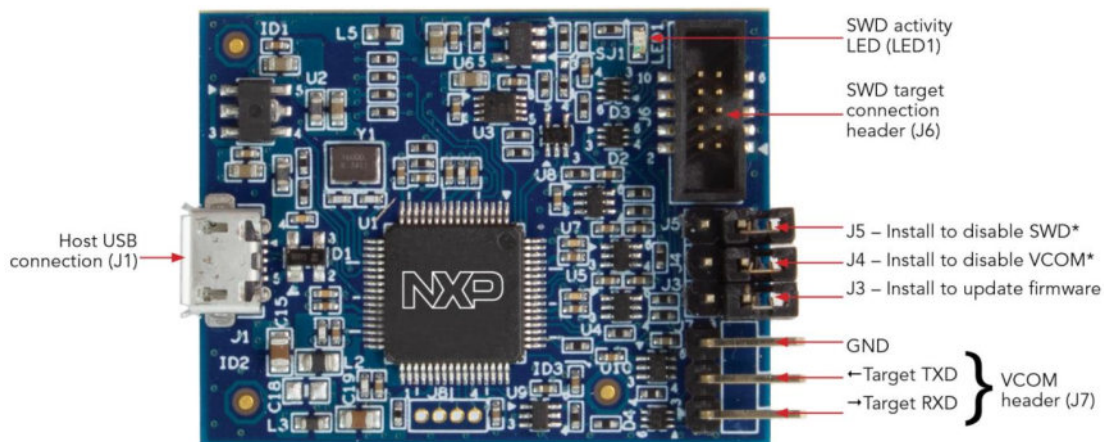


Figure 1 – NXP MCU-LINK Debug/Programmer Probe

5.5 MCU-LINK-PRO Debugger/Programmer Probe (Optional)

The new [NXP MCU-LINK-PRO Debugger Probe](#) provides an alternative debugger probe option with several enhancements (available Sept 2021)

Features in common with MCU-Link:

- High speed USB
- SWD debug
- SWO profiling
- VCOM (USB to UART bridge)

Advanced MCU-Link-Pro features:

- Target energy/power measurement
- USB SPI & I2C bridges for programming/provisioning & host-based application development
- On-board, user-programmable LPC804 for peripheral emulation
- SEGGER J-Link firmware option
- Option to power target system (at 1.8V or 3.3V)
- Hardware capabilities for future enhancements

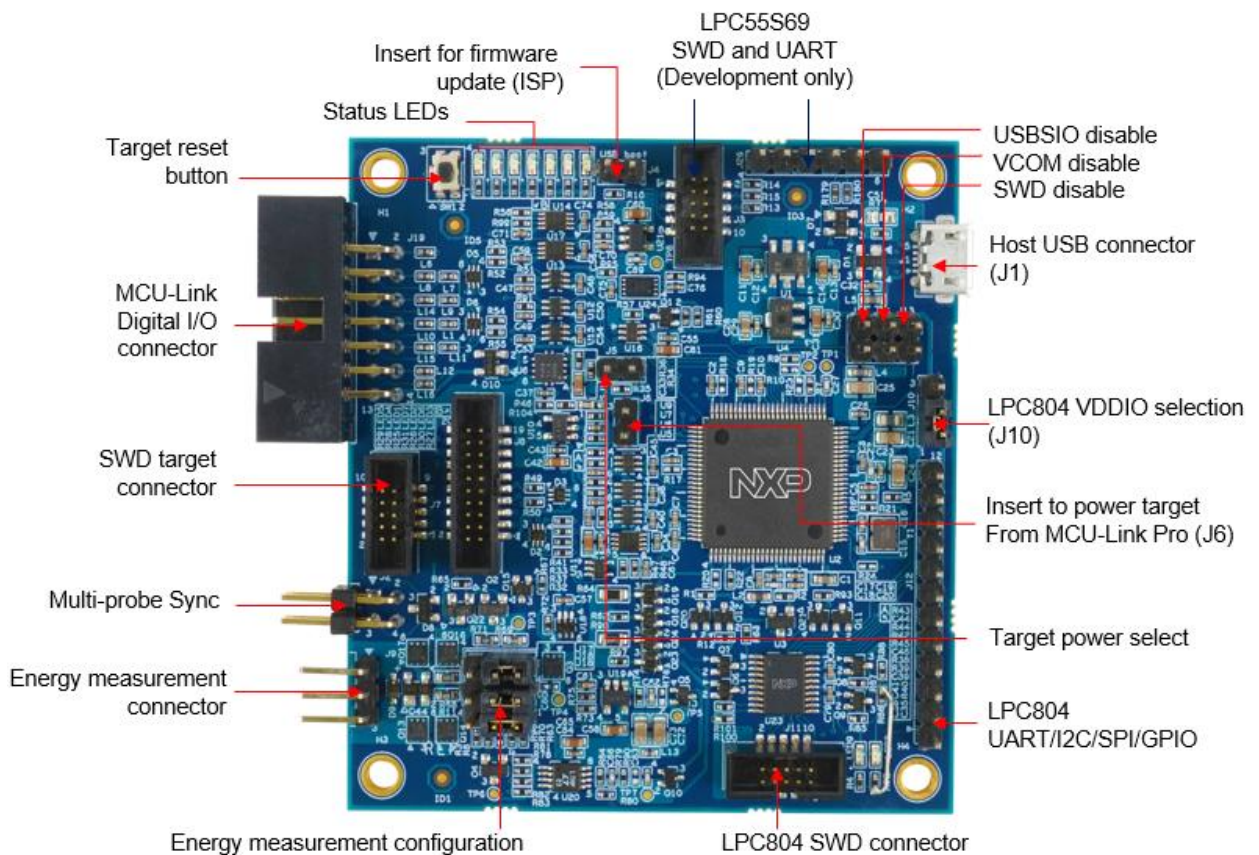
Part# (and link): [MCU-LINK-PRO](#) (MSRP = \$39.99)

Figure 2 – NXP MCU-LINK-PRO Debug/Programmer Probe

5.6 MIPI DSI 7-inch Capacitive Touch LCD Display (Optional)

- Supports up to 1280 x 720 resolution
- Compatible with all MaaXBoard SBC platforms.
- Connects to host via 2-lane MIPI-DSI interface
- Capacitive multi-touch display overlay
- Custom displays available via Avnet Embedded

Part# (and link): [AES-ACC-MAAX-DISP1](#) (MSRP = \$78.95)



5.7 MIPI CSI 5 MP Camera (Optional)

- High quality 5 MP image sensor
- Compatible with all MaaXBoard SBCs and Raspberry Pi
- Attaches to host via 2-lane MIPI CSI ribbon cable
- Supports 1080p30, 720p60 and 640x480p90 video
- Small dimensions (24mm x 25mm x 9mm)

Part# (and link): [AES-ACC-MAAX-CAM1](#) (MSRP = \$26.95)

	
MIPI 7" Display	MIPI 5MP Camera
p/n: AES-ACC-MAAX-DISP1 Available: In stock Price: \$78.95 USD	p/n: AES-ACC-MAAX-CAM1 Available: In stock Price: \$26.95 USD
avnet.me/maax-disp1-buy	avnet.me/maax-cam1-buy

5.8 Other SBCs and SOMs from Avnet Boards (Optional)

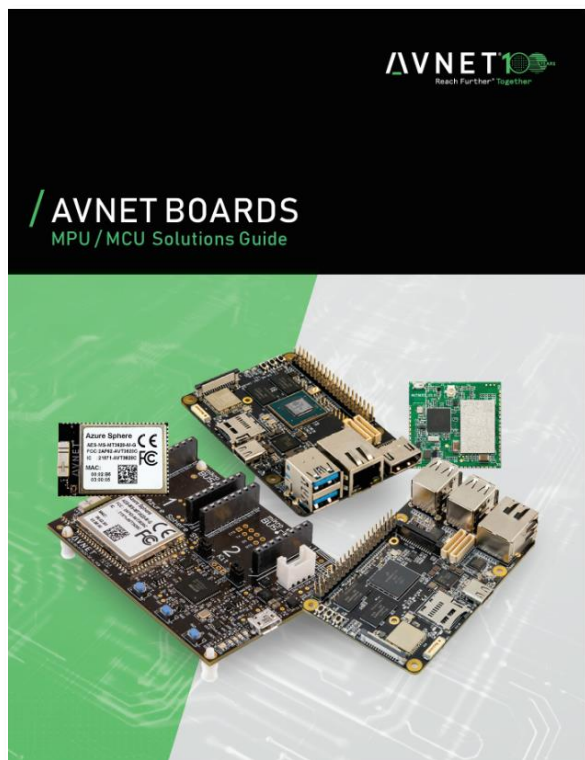
The Avnet **Products & Emerging Technology** engineering team work in close partnership with key suppliers to develop advanced enablement solutions

- Kits / Boards / SOMs / Modules
- Reference Designs
- Trainings / Tutorials / Blogs

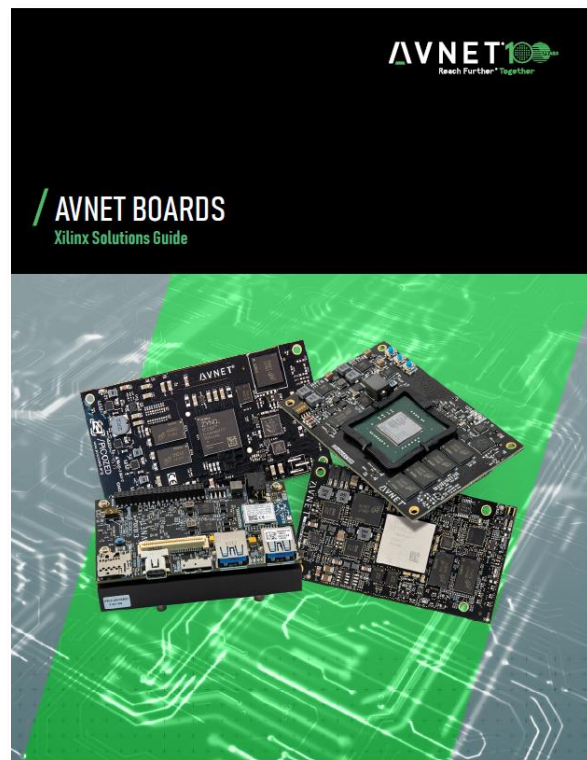


For more information, visit avnet.me/avnetboards

Downloadable Solutions Guides are also available:



avnet.me/mpu-mcu-solutions-guide-2021



avnet.me/xlx-solutions-guide-2021

5.9 Hardware Setup for Application Development

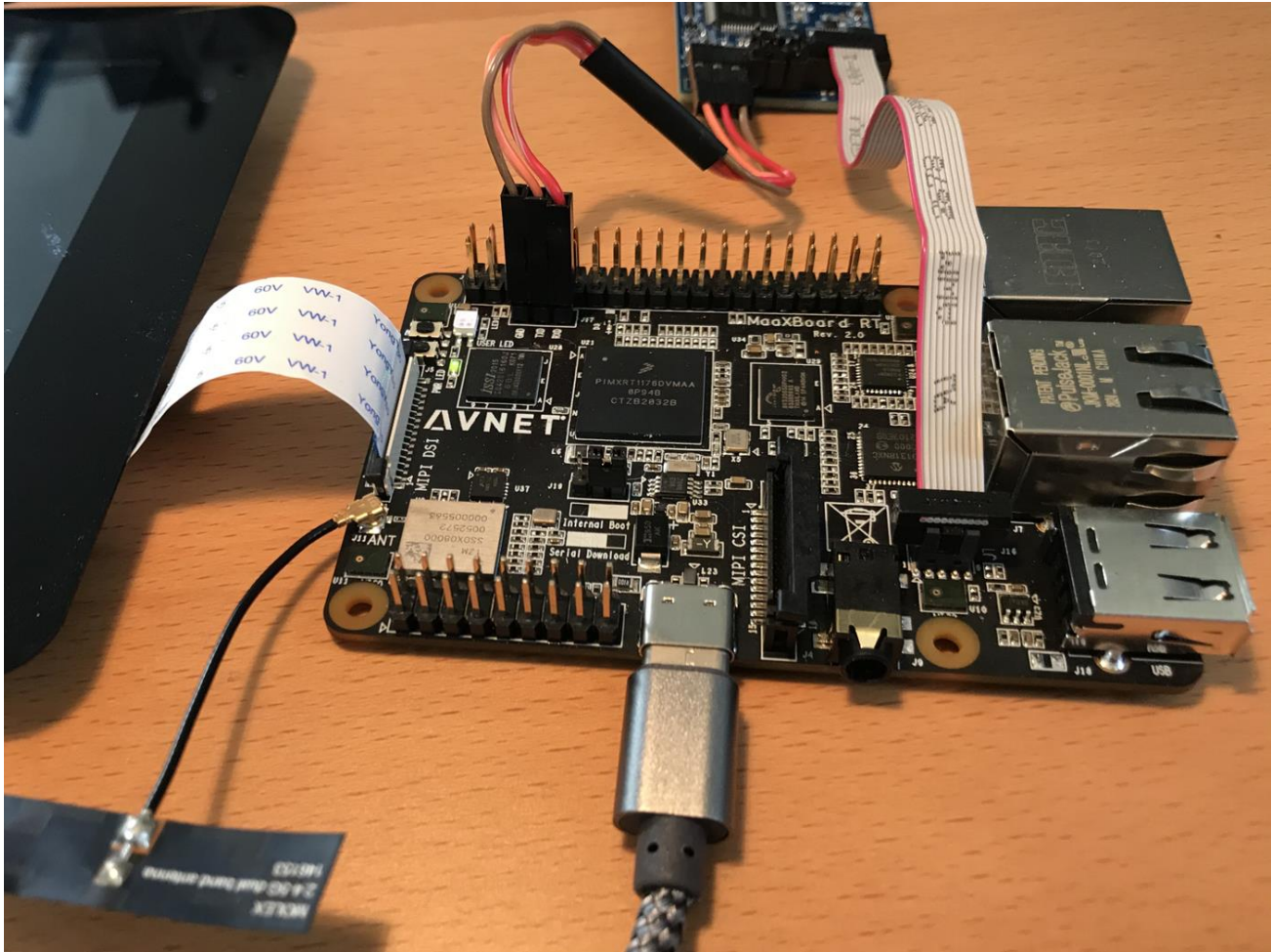


Figure 3 – Typical hardware setup for software development

6 MaaXBoard RT Architecture & Features

6.1 Features

Based on NXP i.MX RT1176 crossover dual-core MCU with advanced real-time capabilities

Cortex-M7 @1GHz

- Class-leading Cortex-M7 benchmarks (2434 DMIPS, 5070 Coremark)
- Super-fast interrupts (12ns latency), up to 220 interrupts sources, 4 interrupt levels
- Exceptional power efficiency (<100uA/MHz), DFVS support for low-power operation.
- High reliability, with ECC support for L1 cache and TCM memory
- Large L1 cache (32KB I-Cache, 32KB D-Cache)
- 512 KB TCM configurable as ITCM or DTCM with Zero-Wait-Cycle Access
- VFPv5 architecture floating point unit, supports single- and double-precision FPU
- Memory protection MPU supports up to 16 memory regions

Cortex-M4 @400 MHz

- Class-leading Cortex-M4 benchmarks (544 DMIPS, 1398 Coremark)
- Fast interrupts (30ns latency), up to 220 interrupts sources, 4 interrupt levels
- Exceptional power efficiency (<30uA/MHz), DFVS support for low-power operation.
- L1 cache (16KB I-Cache, 16KB D-Cache)
- 256 KB TCM with Zero-Wait-Cycle Access

Easy integration into OEM custom products

- Use as a processing sub-assembly in OEM products or as a versatile development board
- Raspberry Pi-4B form-factor facilitates easier enclosure and cabling integration
- 40-pin HAT compatible expansion header (ie. access Pi HAT ecosystem add-on boards)
- 18-pin custom expansion connector

Board is well resourced with high-speed memories

- On-chip: 2 MB RAM, flexible configuration as TCM and OCRAM
- Onboard: 32 MB SDRAM (16 M x 16)
- Onboard: 32 MB HyperFlash (32 M x 8)

Comprehensive set of board interfaces and peripheral devices

- 40-pin Pi-HAT compatible expansion connector
- 18-pin custom expansion connector
- 2x USB host interfaces
- 1x GbE Ethernet (with TSN)
- 1x 10/100 Ethernet (with 1588)
- SWD/JTAG debugger 10-pin mini header (use NXP MCU-Link, or 3rd party debug probe)
- MIPI-DSI touch display interface (2-lane, supports up to 1280x800 display)
- MIPI-CSI camera interface (2-lane, pinout is same as on Raspberry-Pi)
- Audio subsystem with 4x onboard digital microphones plus stereo audio output jack
- 802.11ac Wi-Fi and Bluetooth 5
- U.FL connected external antenna
- High efficiency 5V to 3.3V dc/dc buck convertor, plus low-current low-voltage LDOs
- Operating Temperature: -30~85°C
- Dimensions: 85mm x 56mm (same as Raspberry-Pi 4B)

6.2 Block Diagram – NXP RT1176 Processor

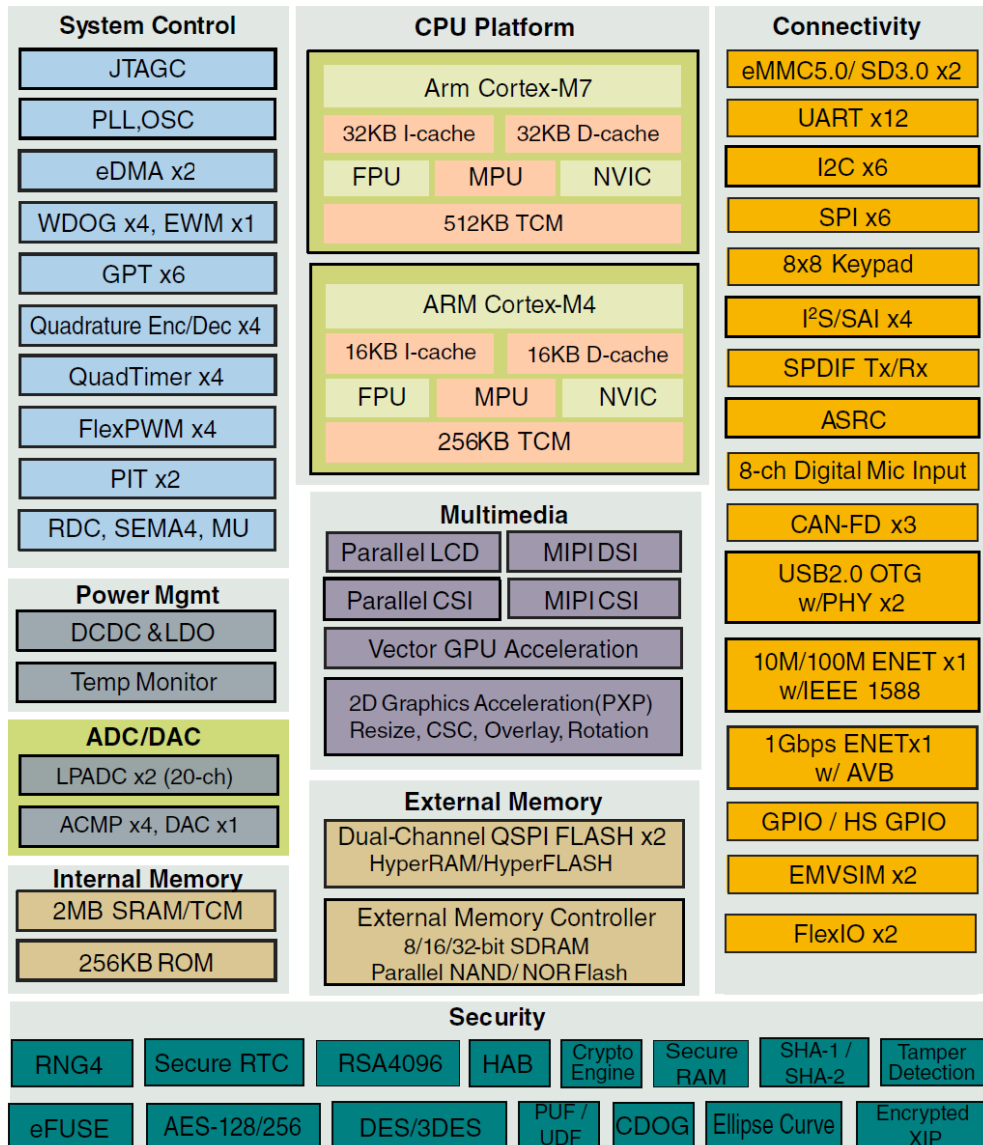


Figure 4 – NXP RT1176 Processor Block Diagram

6.3 Block Diagram – MaaXBoard RT (Rev.3 PCB)

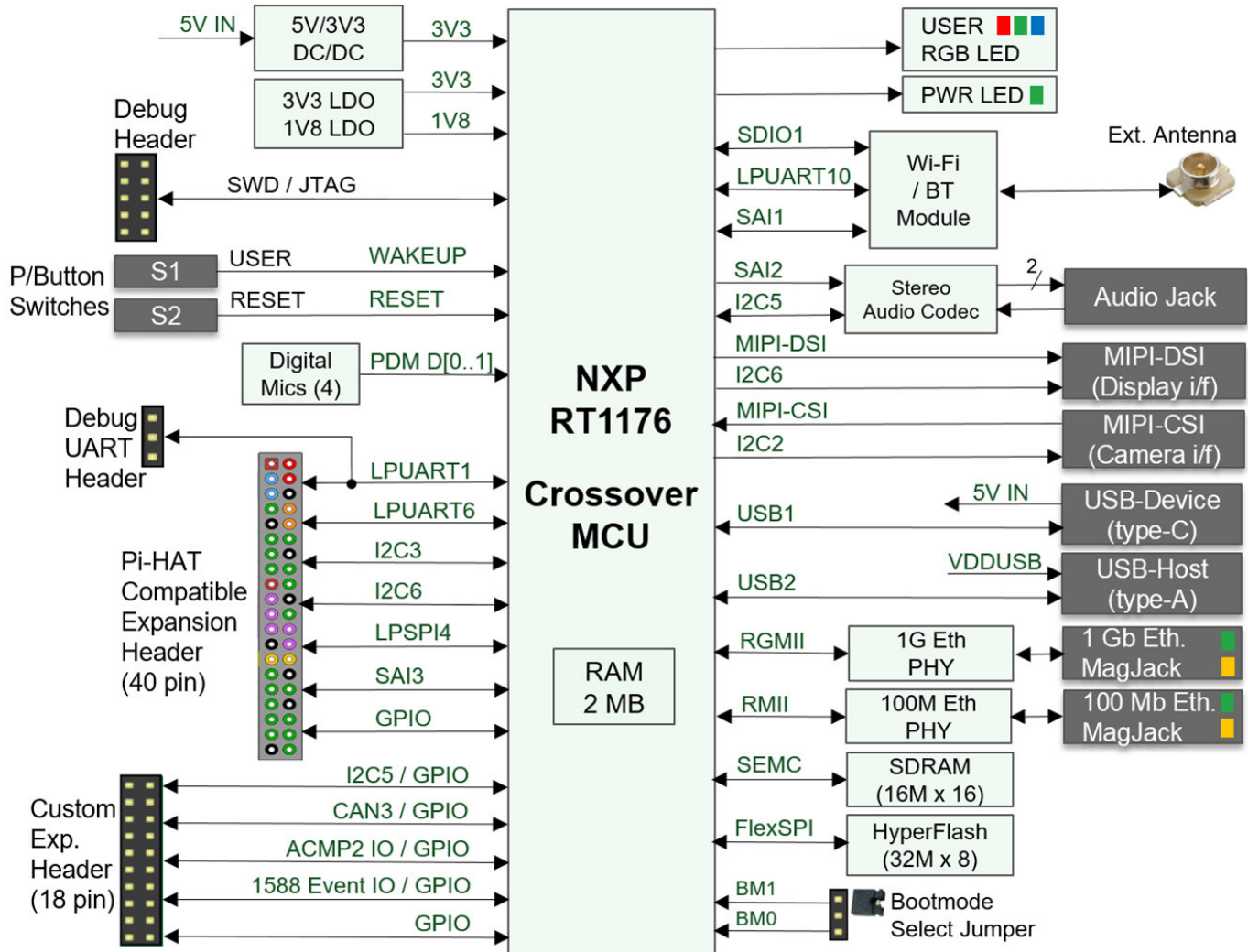
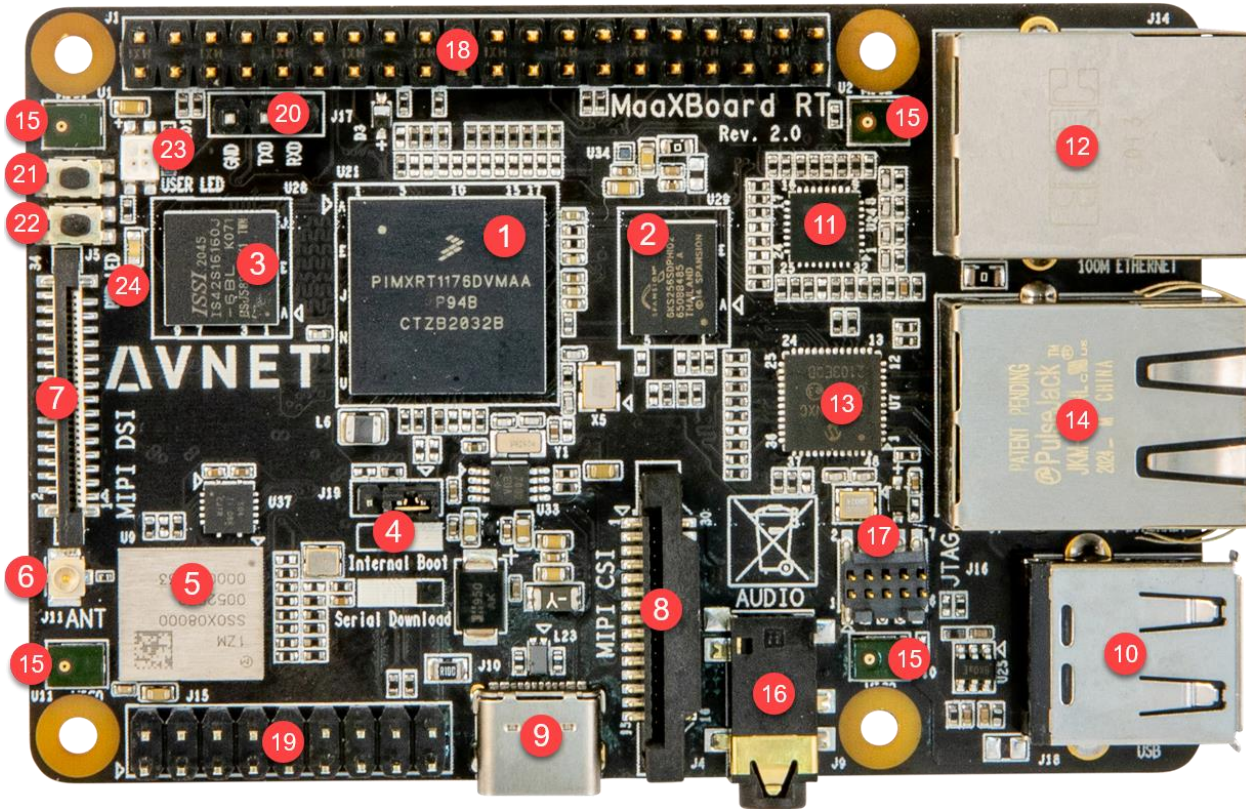


Figure 5 – MaaXBoard RT Block Diagram

6.4 Location of Components on MaaXBoard RT



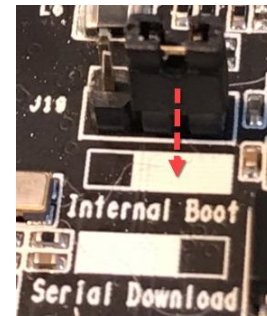
#	Component Description	#	Component Description
1	NXP RT1176DVMAA processor	13	1G Ethernet PHY
2	Spansion HyperFlash memory	14	1G Ethernet MagJack RJ45 connector
3	ISSI SDRAM memory	15	PDM digital microphones (4)
4	Boot Mode Selection Jumper (J19)	16	Stereo audio output jack
5	Dual-band Wi-Fi/BT combo module	17	SWD / JTAG debug mini 10-pin header
6	U.FL external antenna connector	18	Pi HAT compatible 40-pin header (J1)
7	MIPI-DSI touch display interface	19	Custom expansion 18-pin header (J15)
8	MIPI-CSI camera interface	20	Console Debug UART 3-pin Header (J17)
9	USB 2.0 device + power type-C connector	21	USER button switch (S1)
10	USB 2.0 host type-A connector	22	RESET button switch (S2)
11	10/100 Ethernet PHY	23	USER RGB LED
12	10/100 Ethernet MagJack RJ45 connector	24	PWR LED

Table 3 – Location of Key Components on MaaXBoard RT

6.5 Jumpers, LEDs, and Switches

6.5.1 Boot Mode Configuration Jumper (J19)

Link Pins	M1 M0 State	Setting Description
None	00	Boot from Fuses
1-2	10	Internal Boot / IDE-based Development
2-3	01	Serial Downloader (using USB or UART)



6.5.2 Button Switches

Two pushbuttons are located at board edge, between MIPI-DSI connector and 40-pin header

Button Switch	Function	Ref. Des	RT1176 Pin	RT Board Function
USER	WAKEUP (SNVS ANA power domain)	S1	[T8] GPIO13_IO00	GPIO Input / Wakeup
RESET	VDD_3V3 On/Off (SNVS IN power domain)	S2	n/a	System Reset, active low

6.5.3 Status LEDs

Status LEDs	Color	Ref. Des.	RT1176 GPIO Pin	RT Board Function
PWR	Green	D15	n/a	3V3 status
USER-RED	Red	LED1	[R15] GPIO9_IO07	GPIO / PWM
USER-GREEN	Green	LED1	[N3] GPIO8_IO28	GPIO / PWM
USER-BLUE	Blue	LED1	[R17] GPIO9_IO09	GPIO / PWM

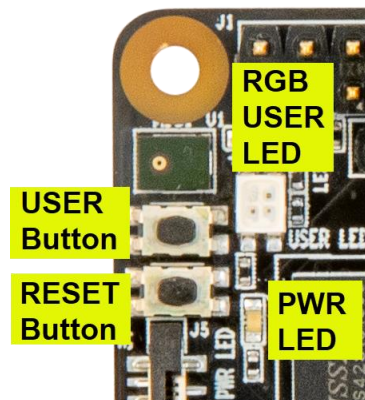


Figure 6 – LEDs and Button Switches

6.6 Memory Resources

The NXP RT1176 device is extremely flexible in what memory is utilized for application storage, application execution and data storage. Note: Significant differences in performance can be expected, when comparing code execution in the different types of memory.

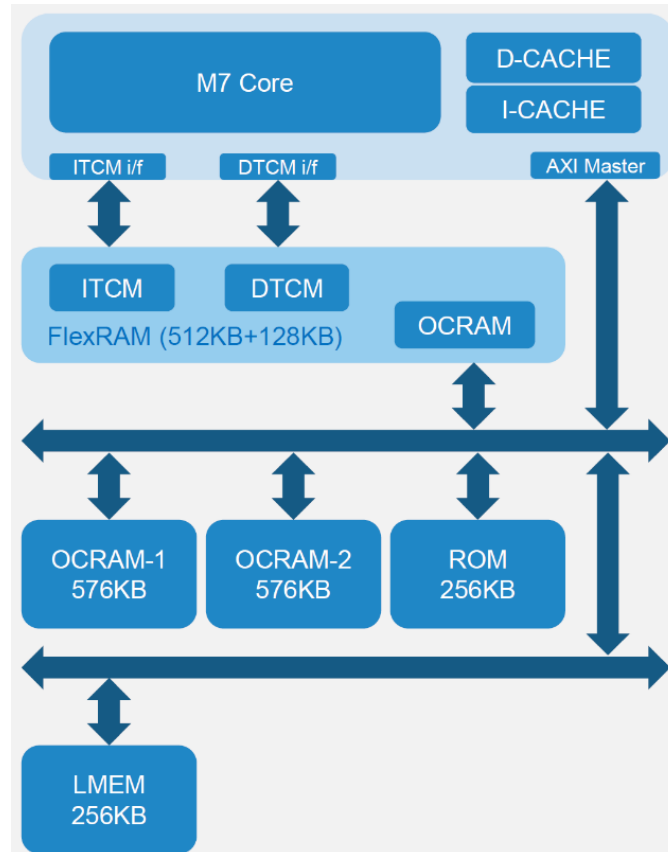


Figure 7 – RT1176 on-chip memory resources

6.6.1 Memory Maps for Application Development

Tabled below are typical memory map settings for applications using RAM and SDRAM

a) RAM-based Cortex-M7 low-latency/high-performance real-time application

Memory Type	Mem. Size	Mem. Width	Mem. Speed	Hex Start Address	Hex Mem. Size
SRAM_ITC_cm7	256 KB	32 bit	TBD	0x0	0x40000
SRAM_DTC_cm7	256 KB	32 bit	TBD	0x20000000	0x40000
SRAM_OC1	512 KB	32 bit	TBD	0x20240000	0x80000
SRAM_OC2	256 KB	32 bit	TBD	0x202c0000	0x40000
NCACHE_REGION (in OCRAM)	256 KB	32 bit	TBD	0x20300000	0x40000
SRAM_OC_ECC1	64 KB	32 bit	TBD	0x20340000	0x10000
SRAM_OC_ECC2	64 KB	32 bit	TBD	0x20350000	0x10000
HyperFlash [bus=FlexSPI] p/n= S26KS256SDPBHI020	32 MB	8 bit	166 MHz (DDR)	0x30000000	0x2000000

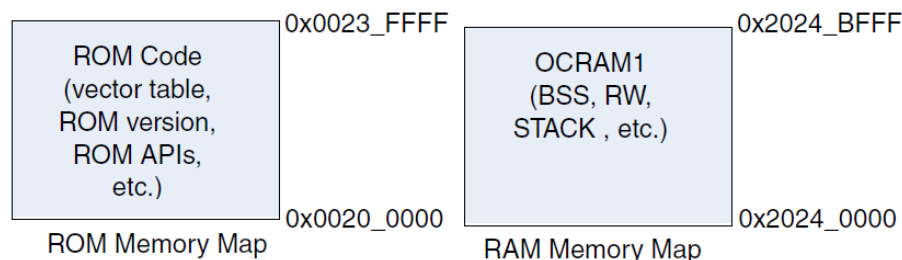
b) SDRAM-based Cortex-M7 application (eg. for an application with GUI display)

Memory Type	Mem. Size	Mem. Width	Mem. Speed	Hex Start Address	Hex Mem. Size
SRAM_ITC_cm7	256 KB	32 bit	TBD	0x0	0x40000
SRAM_DTC_cm7	256 KB	32 bit	TBD	0x20000000	0x40000
SRAM_OC1	512 KB	32 bit	TBD	0x20240000	0x80000
SRAM_OC2	256 KB	32 bit	TBD	0x202c0000	0x40000
SRAM_OC_ECC1	64 KB	32 bit	TBD	0x20340000	0x10000
SRAM_OC_ECC2	64 KB	32 bit	TBD	0x20350000	0x10000
HyperFlash [bus=FlexSPI] p/n= S26KS256SDPBHI020	32 MB	8 bit	166 MHz (DDR)	0x30000000	0x2000000
SDRAM [bus=SEMC0] p/n = IS42S16160J-6BL	32 MB	16 bit	166 MHz (DDR)	0x80000000	0x1000000
NCACHE_REGION p/n = IS42S16160J-6BL	32 MB	16 bit	166 MHz (DDR)	0x81000000	0x1000000

Table 4 – Memory map settings when using RAM and SDRAM

6.6.2 Internal ROM/RAM memory map

The RT1176 has 256 KB of on-chip ROM. (ROM is only used by the M7 core during boot up).



6.6.3 HyperFlash Memory

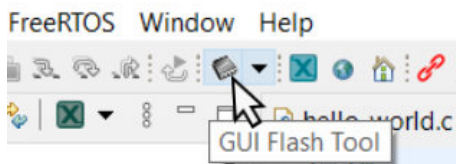
Program storage on MaaXBoard RT, is 32 MB of HyperFlash memory (Cypress/Spansion p/n **S26KS256S**). It's octal DDR data interface ensures fast boot-times and high-performance execute-in-place operation, with more than 4x faster read bandwidth than QSPI NOR flash, at 1/3 the pin-count of parallel NOR flash.

The software examples in the NXP RT1170-EVK SDK package all target QSPI NOR flash memory. When importing an SDK example for use on MaaXBoard RT, you **must** replace the default QSPI driver (MIMXRT1170_SFDP_QSPI.cfx) with the Avnet-provided **MaaXBoard_S26KS256.cfx** flash loader driver.

Initial programming of the HyperFlash “flashloader” into memory

Recommended method to program the HyperFlash flashloader driver into memory is as follows:

- 1) Make sure the **J19** bootmode jumper is in place, in the “**Internal Boot**” position
- 2) Import an example project from the SDK 2.10.1 RT1170-SDK package (e.g., Hello World)
- 3) In the project's **XIP** folder, delete **evkmimxrt1170_flexspi_nor_config.c** and replace this with the Avnet version file (of the same name), which specifies the HyperFlash parameters instead of the QSPI flash configuration parameters used in NXP's SDK examples.
- 4) Into the same **XIP** folder, copy the Avnet-provided **MaaXBoard_S26KS256.cfx** flash loader driver
- 5) With your project selected, go to the **Quick Start** panel, click on **Edit Project Settings**, open **C/C++ Build → MCU Settings**, then click on the default MIMXRT1170_SFDP_QSPI.cfx driver listed at end of the BOARD_FLASH line. Use the button that appears, to browse to and instead select the **MaaXBoard_S26KS256.cfx** file that you just added.
- 6) Add **MaaXBoard_S26KS256.cfx** also as the “**Default LinkServer Flash Driver**”
- 7) From the Menu bar, launch the **GUI Flash Tool**



- 8) In Flash GUI Tool, select the **Erase** tab and make sure the default Flash Driver is **MaaXBoard_S26KS256.cfx** then select **Mass Erase** followed by **Run**.
- 9) If Mass Erase fails on first attempt, power-cycle the board then repeat this Mass-Erase exercise
- 10) Once Mass Erase successfully completes, the board is then ready for debugging or running applications that execute in place in HyperFlash memory.

GUI Flash Tool

GUI Flash Tool for:
MCUXpresso IDE LinkServer (inc. CMSIS-DAP) &
Program file into flash:

Target: MIMXRT1176xxxxx

Probe Options
Probe specific options

Connect script: RT1170_connect_M7_wake_M4.scp Workspace... File System...

Default Flash Driver: **MaaXBoard_S26KS256.cfx** Workspace... File System...

Reset Handling: Default

Flash Reset Handling: Default

Boot ROM Stall:

Wire Speed:

☐ Reset the target on connection ☐ Disable use of preconnect script

Target Operations
Select the target flash operation to perform

Program **Erase**

Algorithm
Select the algorithm for flash erasing

☒ **Mass erase**

☐ Erase by sector

☐ Check blank

General Options
Flash programming tool options

Additional options:

☐ Repeat on completion ☒ Enable flash hashing ☐ Preview command

☒ Clear console

Run... Cancel

6.6.4 HyperFlash Programming using NXP MCUBootUtility Application

Besides normal programming via the debugger probe and the SWD interface, the hyperflash can also be programmed via the USB-C interface or a UART interface, using “Serial Download” boot mode.

To select this mode, the **J19** bootmode jumper needs to be moved to the “**Serial Download**” position.

The [NXP-MCUBootUtility](#) PC application is a user-friendly GUI based tool for flash erase and programming, using this mode, either via the board’s USB-C connector, or the debug UART 3-pin header.

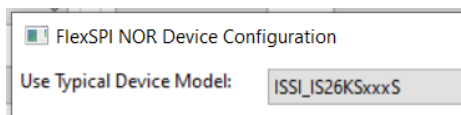
In the event that HyperFlash memory gets corrupted during application development, then this tool (NXP-MCUBootUtility) **must** be used to erase the flash memory.

The short procedure below, is detailed under the “Boot Utility” section of the following online blog article: <https://mcuoneclipse.com/2019/01/02/regaining-debug-access-of-nxp-i-mx-rt1064-evk-executing-wfi/> or in the ReadMe instructions at: <https://github.com/JayHeng/NXP-MCUBootUtility/>

Procedure to erase- and restore flashloader driver (MaaXBoard_S26KS256.cfx) in HyperFlash:

Note: Steps 3 – 13 steps on the GUI interface are shown on the next page...

- 1) Powerdown the board and set jumper **J19** to the **Serial Download** position (bridge pins 2-3)
- 2) Power-up the board again, preferably using USB-C connected power and data from the development PC to the board’s type-C USB connector (USB-HID provides faster operation than UART mode)
- 3) Set **i.MXRT117x** as the target device
- 4) Set **ISSI_IS26KSxxxS** as Typical Device Model in the Boot Device Configuration



- 5) De-select the “**One-Step**” check-box
- 6) Click on **Connect to ROM** button
- 7) Click on **Connect to FlashLoader** button
- 8) Click on **Configure Boot Device** button
- 9) Now click on the right-most tabbed view **Boot Device Memory**
- 10) Enter **0x30000000** for **Start / Offset** address of the Flash memory,
- 11) Enter **0x400000** for **Byte Length (For Read/Erase)** of the Flash memory (ie. 256 KB)
- 12) Click on **Erase** to clear the flashloader from this range of flash memory
- 13) The “Log” Window should not report any errors
(*Read button can be used inspect erased memory, but set a shorter Byte Length before doing that!!*)
- 14) Once flash memory has been completely erased by this tool, the board is then powered-down
- 15) Place bootmode jumper **J19** back in the Internal Boot (IDE development) position (**pins 1-2**)
- 16) Now power-up the board again (from wall-wart or PC as the 5V power source).
- 17) In MCUXpresso IDE, the **GUI Flash Tool** mass-erase should now be run (described in previous section) to reload the default **MaaXBoard_S26KS256.cfx** flashloader driver back into flash memory.

NXP MCU Boot Utility v3.2.0

File Edit View Tools Window Help

Target Setup

MCU Series: i.MXRT

MCU Device: **3** i.MXRT117x

Boot Device: FLEXSPI NOR

4 Boot Device Configuration

Device Configuration Data (DCD)

Port Setup

☐ UART ☒ USB-HID

Vendor ID: 0x15A2

Product ID: 0x0073

5 ☐ One Step

6 7 8 **Connect to Flashloader**

Device Status

-----FlexSPI NOR memory-----
 Page Size = -----
 Sector Size = -----
 Block Size = -----
 Page Size = 512 Bytes
 Sector Size = 256.0 KB
 Block Size = 256.0 KB
 -----MCU device Register-----
 HAB status = Open

Secure Boot Type DEV Unsigned Image Boot **All-In-One Action** **NXP**

Image Generation Sequence **Image Loading Sequence** **eFuse Operation Utility** **Boot Device Memory** **9**

Start / Offset: 0x30000000 **10** Byte Length (For Read/Erase): 0x400000 **11** Bin File: **Browse**

Read **Erase** **12** **Write (Auto Erase)** **Execute From Start**

View Bootable Image **Clear The Screen** ☐ Save image/data file to **Browse**

Log **13**

```
property 17 0
Executing E:\NXP\NXP-MCUBootUtility-master\tools\blhost2_3\win\blhost -t 5242000 -u 0x1FC9,0x013D -j -- load-
image E:\NXP\NXP-MCUBootUtility-master\src\targets\MIMXRT1176\ivt_flashloader.bin
```

00:00.832 

Figure 8 – NXP MCU Boot Utility Settings for Erase of Flashloader

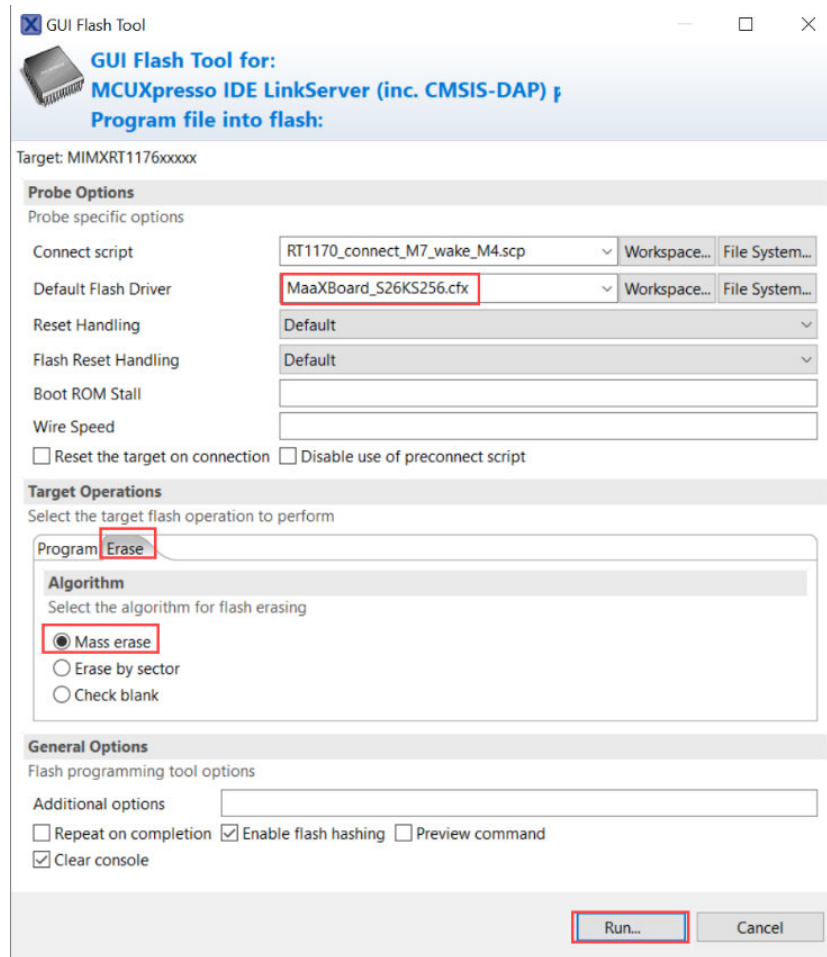


Figure 9 – Mass Erase / Reload of Default Flash Driver

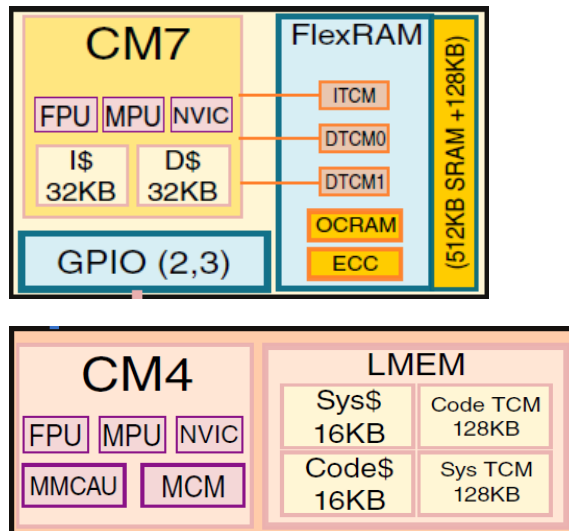
6.6.5 HyperFlash Programming using MCUXpresso Secure Provisioning Tools

Flash programming is also supported via the NXP **MCUXpresso Secure Provisioning Tools** standalone application. This is not however a suitable tool to use during program development.

6.6.6 On-chip TCM RAM and OCRM

The i.MX RT1176 has a total of 2MB of on-chip RAM. This includes 512KB FlexRAM which can be flexibly configured as TCM or general-purpose OCRM (for the M7 core).

Highest performance is achieved when critical application code is executed in TCM memory.



6.6.7 SDRAM

Data storage on MaaXBoard RT, is a single 32MB SDRAM memory device, with 16bit parallel interface (ISSI p/n **S26KS256S**).

When building an application that uses SDRAM, it is of critical importance that the project's **dcd.c** file be replaced by the version of this file that is supplied by Avnet. This is necessary as MaaXBoard RT has x16 width SDRAM interface, while the NXP RT1170-EVK board has a x32 width parallel data interface)

Download the Avnet version **dcd.c** file from:

<http://avnet.me/MaaXBoard-RT-SDK>

6.7 Wireless Connectivity

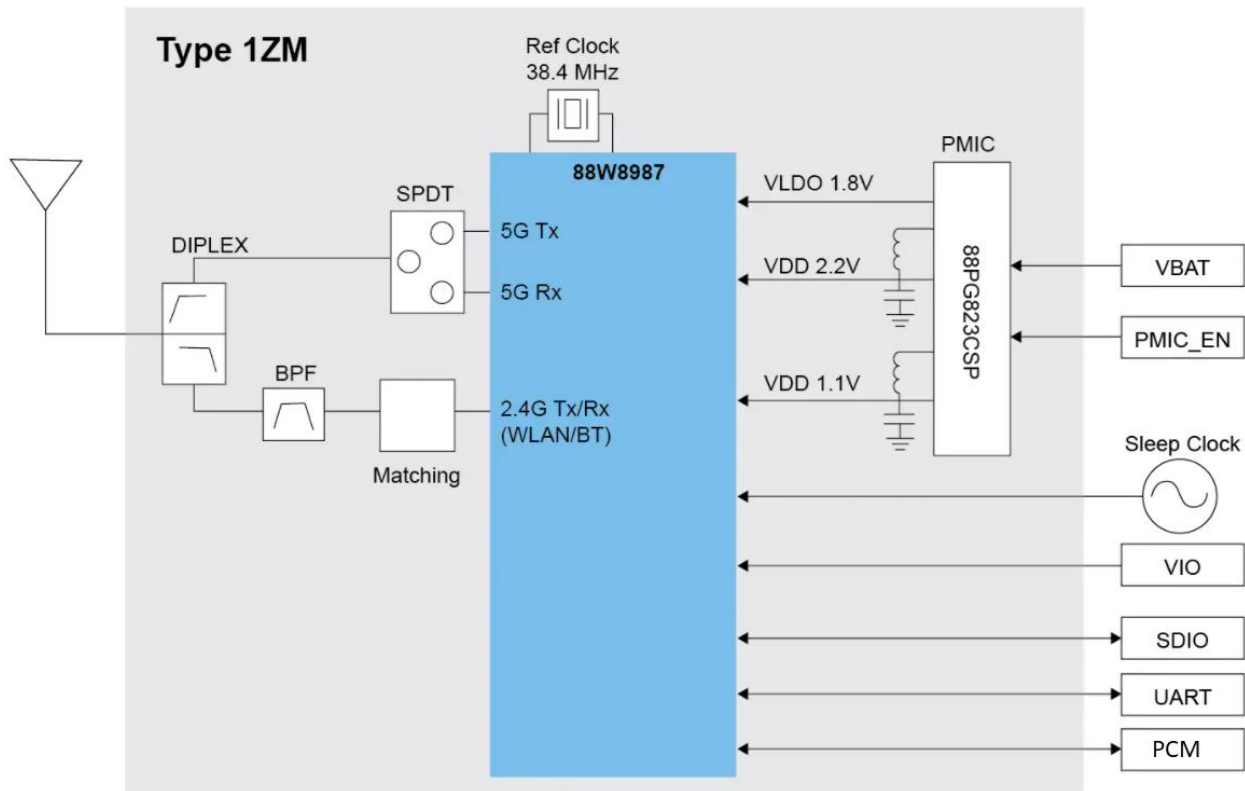


Figure 10 – Murata Type-1ZM Wi-Fi/BT Combo Module Block Diagram

MaaXBoard RT uses a **Murata Type-1ZM** Wi-Fi 5 and Bluetooth 5 combo module, which is based on the **NXP 88W8987** device. Both 2.4 GHz and 5 GHz Wi-Fi operation is supported

Interfaces between the NXP 88W8987 based module and the NXP RT1176 host processor include:

- Wi-Fi SDIO 3.0 (4-bit) interface (uses **SDIO1**)
- BT/BLE UART 4-wire interface (uses **LPUART10**)
- Bluetooth audio PCM interface (uses **SAI1**)

1ZM Module Pin Name	RT Peripheral Resource (on MaaXBoard)	NXP RT1170-EVK SDK Signal Name	NXP RT1170-EVK M.2 Signal Name	RT Peripheral Resource and Pin# (on EVK)
BOARD_InitUSDHCPins <i>USDHC1</i>				
SD_CLK	SD1_CLK [D15]	WIFI_SD_CLK	SDIO_CLK	GPIO_SD_B1_01 [D15]
SD_CMD	SD1_CMD [B16]	WIFI_SD_CMD	SDIO_CMD	GPIO_SD_B1_00 [B16]
SD_DAT0	SD1_DATA0 [C15]	WIFI_SD_DAT0	SDIO_DATA0	GPIO_SD_B1_02 [C15]
SD_DAT1	SD1_DATA1 [B17]	WIFI_SD_DAT1	SDIO_DATA1	GPIO_SD_B1_03 [B17]
SD_DAT2	SD1_DATA2 [B15]	WIFI_SD_DAT2	SDIO_DATA2	GPIO_SD_B1_04 [B15]
SD_DAT3	SD1_DATA3 [A16]	WIFI_SD_DAT3	SDIO_DATA3	GPIO_SD_B1_05 [A16]
	n/a	SD_PWREN_B	SD_PWREN_B	GPIO_AD_35 [G17]
	n/a	SD1_CD_B		GPIO_AD_32 [K16]
	n/a	SD1_VSELECT	VSELECT	GPIO_AD_34 [J16]
BOARD_InitM2WifiResetPins				
PMIC_EN (WL_REG_ON)	GPIO_AD_14 GPIO9_IO13 [N14]	WIFI_RST_B_1V8	/SDIO_RST	GPIO_AD_16 GPIO9_IO15 [N17]
BOARD_InitUSDHCPins				
	n/a	SD1_VSELECT	VSELECT	GPIO_AD_34 [J16]
BOARD_InitM2UARTPins				
MD_UART_RXD	LPUART10 [K16]	MD_UART_RXD	UART_RXD	LPUART2 [A6]
MD_UART_TXD	LPUART10 [H17]	MD_UART_TXD	UART_TXD	LPUART2 [D9]
MD_UART_CTS	LPUART10 [G17]	MD_UART_CTS	UART_CTS	LPUART2 [B6]
MD_UART_RTS	LPUART10 [J16]	MD_UART_RTS	UART_RTS	LPUART2 [A5]
UNDEFINED				
	n/a	WIFI_WAKE_B_1V8 WL_HOST_WAKE	/SDIO_WAKE /PERST1	GPIO_AD_29 [M17]
	n/a	BT_WAKE_3V3	/UART_WAKE	GPIO_AD_27 [N16]

Table 5 – Wi-Fi/BT Module Interface (mapped to SDK/EVK signal names)

Page 29

6.7.1 Wi-Fi SDIO Interface

USDHC1 interface is configured for 1.8V operation and is clocked at 200 MHz
(This is the same SDIO interface as used on the RT1170-EVK)

6.7.2 BT/BLE UART Interface

LPUART10 including hardware flow is routed from the RT1176 via a 3.3V to 1.8V level-shifter, to the 1ZM module's Bluetooth UART interface. (Note uses of LPUART10 interface is different from what used on the RT1170-EVK, which uses LPUART2 for this interface)

6.7.3 BT PCM Audio Interface

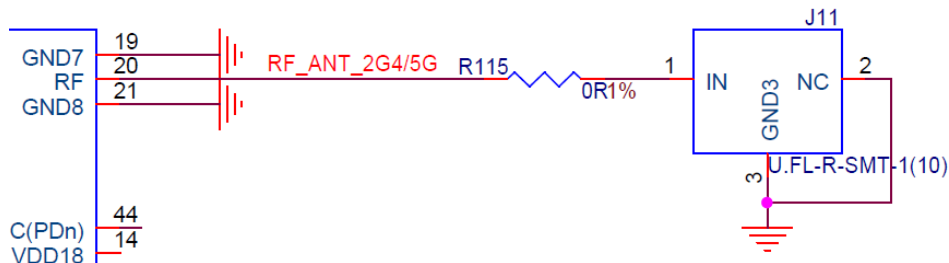
SAI1 is routed from from the RT1176 via a 3.3V to 1.8V level-shifter, to the Murata 1ZM module's Bluetooth UART PCM audio interface. (This use of **SAI1** interface is identical to what is used for this function on the RT1170-EVK)

1ZM Module Pin Name	RT Peripheral Resource and Pin# (on MaaXBoard)	NXP RT1170-EVK SDK Signal Name	NXP EVK M.2 Signal Name	RT Peripheral Resource and Pin# (on EVK)
BOARD_InitPins SAI1				
BT_PCM_DIN	SAI1_TXD [K14]	BT_PCM_TXD	I2S_SD_IN	SAI1_TXD[0] [K14]
BT_PCM_DOUT	SAI1_RXD [K13]	BT_PCM_RXD	I2S_SD_OUT	SAI1_RXD[0] [K13]
BT_PCM_CLK	SAI1_TX_BCLK [K12]	BT_PCM_BCLK	I2S_SCK	SAI1_TX_BCLK [K12]
BT_PCM_SYNC	SAI1_TX_SYNC [J12]	BT_PCM_SYNC	I2S_WS	SAI1_TX_SYNC [J12]

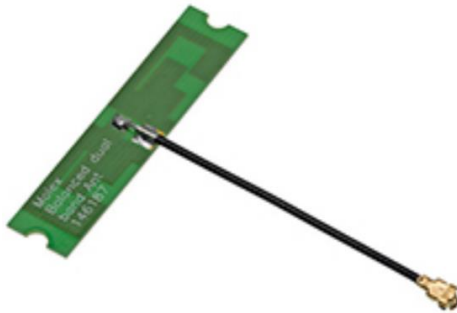
Table 6 – BT PCM Audio Interface (mapped to SDK/EVK signal names)

6.7.4 Wi-Fi / BT antenna

This board only supports an external antenna, attached via UFL / IPEX MHF connector.



The antenna shipped with MaaXBoard RT (as used for regulatory certifications) is [Molex p/n: 1461870050](#)



Operating Frequency	Antenna Gain
2402MHz-2480MHz	3.20dBi
2412MHz~2472MHz	3.20dBi
5180MHz-5240MHz	4.25dBi
5745MHz-5825MHz	4.25dBi
5260MHz-5320MHz	4.25dBi
5500MHz-5700MHz	4.25dBi

6.8 Peripheral Devices and Interfaces

6.8.1 USB 2.0 Host Interface

USB1 controller is assigned to the type-A connector **USB Host** interface

6.8.2 USB 2.0 Device Interface

USB0 controller is assigned to the type-C connector **USB Device** interface

6.8.3 GbE Ethernet (with TSN time-sync.)

The 1G Ethernet subsystem is comprised of:

- **ENET_1G** RT1176 Gigabit Ethernet MAC peripheral,
- MicroChip KSZ9131RNXCA Ethernet transceiver (U7) with RGMII interface,
- RJ45 (J7) with integrated magnetics.

6.8.4 10/100 Ethernet (with IEEE 1588 time-sync.)

The 10/100 Ethernet subsystem is comprised of:

- **ENET** RT1176 Ethernet MAC peripheral,
- Microchip KSZ8081RNBCE Ethernet transceiver (U24) with RMII interface,
- RJ45 (J14) with integrated magnetics.

6.8.5 CAN-FD

CAN3 TXD and RXD signals are pinned-out directly to pins on the 18-pin Expansion connector. To interface this FlexCAN peripheral with external CAN devices, requires the addition of a CAN transceiver and suitable ESD protection

6.8.6 MIPI-DSI Display and Touchscreen

2-lane, supports up to 1280x720 display. This has same 30-pin MIPI connector form-factor as on Raspberry Pi but has customized pinout and includes I2C touchscreen controller interface

6.8.7 MIPI-CSI Camera

2-lane, the pinout is same as used on Raspberry-Pi boards. This optional 5 MP MIPI-CSI camera is based on same OV5640 image sensor as used with the NXP RT1170-EVK board.

6.8.8 Digital Microphones

Four PDM digital microphones (ST p/n: MP34DT05-A) are located adjacent to the four mounting holes on MaaXBoard RT. These are muxed onto two processor inputs:

PDM Microphones	PDM Data Input
MIC0, MIC2	PDM_DATA_0
MIC1, MIC3	PDM_DATA_1

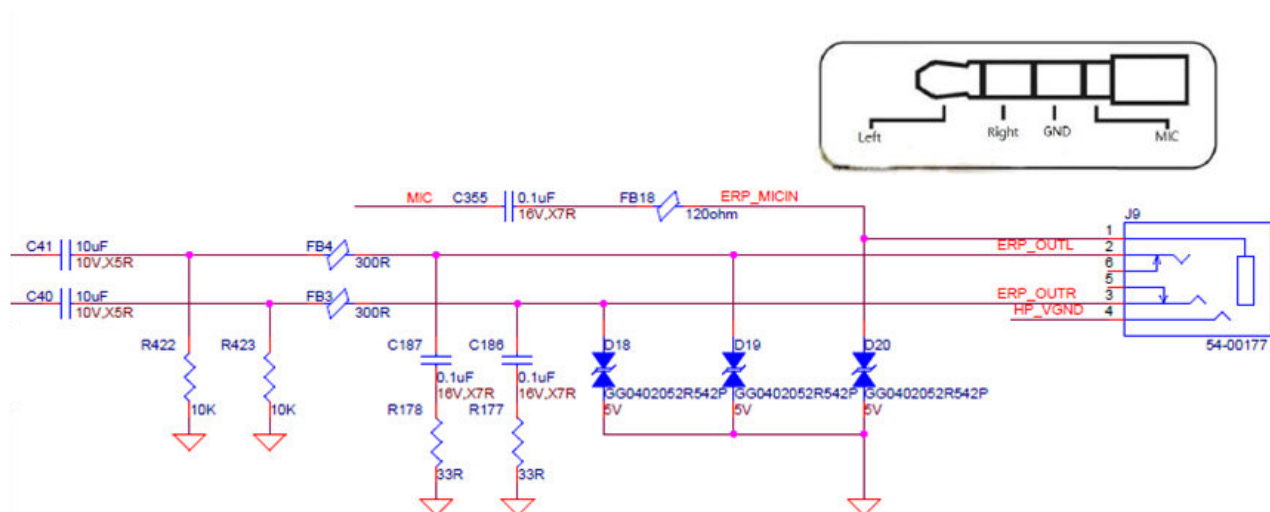
6.8.9 Audio Codec

SAI2 interface of RT1176 (Serial Audio Interface #2) is connected to the NXP SGTL5000 stereo audio Codec. PCM audio between Wi-Fi/BT module and the Codec, requires routing through the RT1176 processor.

Note! Use of SAI2 and a different Codec means that implementation of audio applications is different to that of the SDK examples targeting the RT1170 EVK.
(Refer to Ref.Design #1 for guidance)

SAI1 (Serial Audio Interface #1) connects the RT1176 processor with the Bluetooth PCM audio interface of the WiFi/BT module.

6.8.10 J9: Stereo Audio Jack



6.8.11 J1: Pi-HAT compatible 40-pin header



Possible HAT Pin Function	MaaXBoard RT Signal Name	Odd Pin #	Even Pin #	MaaXBoard RT Signal Name	Possible HAT Pin Function
3V3	VDD_3V3	1	2	5V_SYS	5V
I2C_SDA	I2C3_SDA	3	4	5V_SYS	5V
I2C_SCL	I2C3_SCL	5	6	GND	GND
GPIO / GPCLK0	GPIO8_IO27	7	8	LPUART1_TXD	UART_TX
GND	GND	9	10	LPUART1_RXD	UART_RX
GPIO / RTS	GPIO9_IO12 ⁽¹⁾	11	12	SAI3_TX_BCLK	PCM_CLK
GPIO	GPIO9_IO18	13	14	GND	GND
GPIO	GPIO9_IO27	15	16	LPUART6_RXD	GPIO
3V3	VDD_3V3	17	18	LPUART6_TXD	GPIO
SPI_MOSI	LPSPi4_SDI	19	20	GND	GND
SPI_MISO	LPSPi4_SDO	21	22	GPIO9_IO30	GPIO
SPI_SCLK	LPSPi4_SCK	23	24	LPSPi4_CS0	SPI_CS0
GND	GND	25	26	GPIO8_IO16	SPI_CS1
EEPROM_SDA	I2C6_SDA ⁽²⁾	27	28	I2C6_SCL ⁽³⁾	EEPROM_SCL
GPIO / GPCLK1	GPIO9_IO25	29	30	GND	GND
GPIO / GPCLK2	GPIO9_IO26	31	32	GPIO9_IO15 ⁽⁴⁾	GPIO / PWM0
GPIO / PWM1	GPIO9_IO02	33	34	GND	GND
GPIO / PCM_FS	SAI3_TX_SYNC	35	36	GPIO2_IO15	GPIO / CTS
GPIO	GPIO8_IO12	37	38	SAI3_RXD	GPIO / PCM_DIN
GND	GND	39	40	SAI3_TXD	GPIO / PCM_DOUT

Table 7 – Pi-HAT compatible 40-pin header (J1)

Notes

- 1) **Pin 11** (GPIO9_IO12) - connector I/O by default (R168 is unpopulated)
- 2) **Pin 27** (I2C6_SDA) - connector I2C6 is shared with touch-screen I2C6 interface
- 3) **Pin 28** (I2C6_SCL) - connector I2C6 is shared with touch-screen I2C6 interface
- 4) **Pin 32** (GPIO9_IO15) - connector I/O not available if 1G Ethernet in use (MII0_MDC)

6.8.12 Pi HAT Expansion Boards

- Strong ecosystem of Pi HAT boards support a wide range of functionality
- See listings at websites such as <https://pinout.xyz/boards>
- Height of stacked boards is minimized as RT1176 does not require heatsink

6.8.13 MikroE Click Boards

- Over 1000+ Click Boards available (orderable from Avnet)
- Inexpensive Pi HAT adapters available (for 2 Click boards)
- Parametric search tool on [MikroE website](https://www.mikroe.com)
- Open source library code available at <https://www.mikroe.com/click-boards>

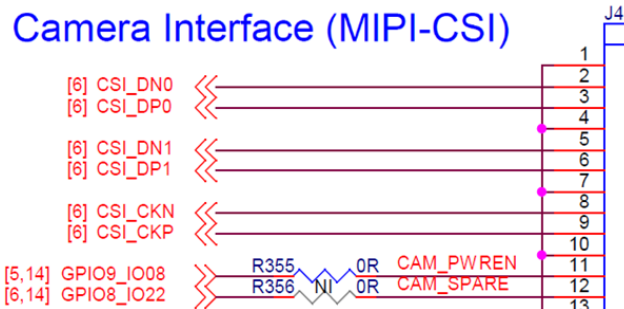
6.8.14 J15: Custom 18-pin expansion header

MaaXBoard RT Port Name	MaaXBoard RT Signal Name	Odd Pin #	Even Pin #	MaaXBoard RT Signal Name	MaaXBoard RT Port Name
VDD_3V3	VDD_3V3	1	2	VDD_3V3	VDD_3V3
GPIO_LPSR_00	CAN3_TX	3	4	GND	GND
GPIO_AD_01	1588_1G_EVENT1_OUT	5	6	CAN3_RX	GPIO_LPSR_01
GPIO_AD_07	1588_EVENT1_OUT	7	8	1588_1G_EVENT1_IN	GPIO_AD_00
GPIO_EMC_B2_12	GPIO8_IO22 ⁽¹⁾	9	10	1588_EVENT1_IN	GPIO_AD_06
GND	GND	11	12	ACMP2_OUT	GPIO_AD_18
GPIO_AD_09	GPIO9_IO08 ⁽²⁾	13	14	ACMP2_IN1	GPIO_AD_04
GPIO_LPSR_05	I2C5_SCL ⁽³⁾	15	16	GND	GND
GND	GND	17	18	I2C5_SDA ⁽⁴⁾	GPIO_LPSR_04

Table 8 – Custom 18-pin Expansion header (J15)

Notes:

- Pin 9** (GPIO8_IO22) - connector I/O by default (R168 unpopulated, otherwise used for CAM_SPARE). A subset of cameras may require use of **CAM_SPARE** pin on the CSI camera interface. In those cases **Pin 9** of **J15** 18-pin header may not be used as connector I/O
- Pin 13** (GPIO9_IO08) - **CAM_PWREN** by default (R356 populated, otherwise use for connector I/O). In applications using CSI camera, **pin 13** of **J15** 18-pin header may not be used as connector I/O
- Pin 15** (I2C5_SCL) - connector I2C5 is shared with Codec I2C5 interface
- Pin 18** (I2C5_SDA) - connector I2C5 is shared with Codec I2C5 interface



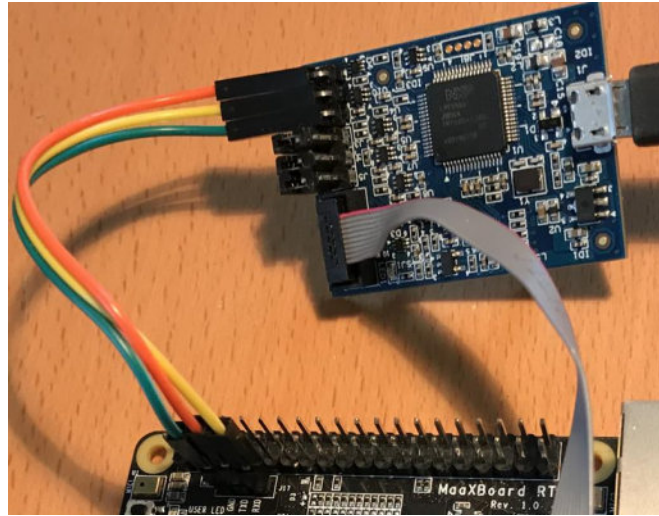
6.8.15 J17: UART1 USB Serial Port 3-pin header

J17 Pin	Label on MaaXBoard RT	Label on MCU-LINK
1	GND	GND
2	TXD	TX to Target
4	RXD	RX to Target

Table 9 – UART1 console debug VCOM 3-pin header (J17)

Note:

- The TXD and RXD wire-order is crossed-over between MCU-LINK debugger and MaaXBoard RT
- The UART1 TXD and RXD signals are also exposed on the 40-pin Pi HAT header



6.8.16 SWD/JTAG debugger 10-pin mini-header

- The 10-pin Mini-header by default supports the SWD interface tabled below

JTAG	MCU-LINK SWD	Pin #	Pin #	MCU-LINK SWD	JTAG
JTAG_TMS	SWD_IO	2	1	IF_VREF	VDD_3V3
JTAG_TCK	SWCLK	4	3	GND	GND
JTAG_TDO	SWO	6	5	GND	GND
JTAG_TDI	IF_TDI	8	7	IF_ISPEN	GND*
nRST	IF_RST	10	9	IF_DETECT	JTAG_nTRST*

Table 10 – SWD/JTAG debugger 10-pin mini-header (J16)

***Note:** For JTAG interface: Reconfigure pin 7 and pin 9 PCB bridges to support these signals

6.9 Power Input, Protection and Regulation

6.9.1 USB type-C Connector

The USB type-C connector is used for 5V power and provides a USB-Device interface

6.9.2 ESD Protection

All USB connectors have high-speed ESD protection on their power rails and data lines

6.9.3 Power Regulation

A 5V to 3.3V dc/dc buck convertor regulates the Vcc rail voltage (rated @ 3A max)

6.9.4 Measuring Power Consumption

An inexpensive current-measurement USB power-measurement dongle is recommended during development, in-line with the USB connection to the host computer, for monitoring 5V input current draw. The USB dongle meter shown here can accommodate a type-C or type-A cable connections and is available online for around \$20

<https://www.amazon.com/Tester-Eversame-Voltmeter-Ammeter-Braided/dp/B07MGQZHGM>

Note: An invalid current measurement will be seen if MaaXBoard RT is powered from a USB port from the same PC as the debugger probe!

To achieve a useful current measurement, the board must be powered from a separate power-adaptor, or the debugger probe must be fully disconnected.



Figure 12 – Current-measurement USB Dongle

Software Enablement

7 Development Software Installation

7.1 NXP MCUXpresso IDE

Download and install the latest copy of MCUXpresso IDE from the following NXP site:
<https://www.nxp.com/mcuxpresso>

7.2 NXP MCUXpresso SDK

Developers can configure and download the latest SDK for the RT1170-EVK from the NXP SDK Builder website, that is accessed at <https://mcuxpresso.nxp.com/en/select>

Use “search by name” for: “RT1170-EVK”MIMXRT1170-EVK
then click **Build MCUXpresso SDK_2.10.1** to download...

To install, simply drag & drop the SDK zip file into the **Installed SDKs** panel of MCUXpresso IDE

- **SDK_2.10.0_MIMXRT1170-EVK.zip**

Four files defined by Avnet also need to be downloaded from the following sharepoint page:
<http://avnet.me/MaaXBoard-RT-SDK>

- **MaaXBoard_S26KS256.cfx**
- **evkmimxrt1170_flexspi_nor_config.c**
- **dcd.c**
- **board.h**

7.3 NXP GUI Guider

If using the optional 7" MIPI LCD graphical display, it is recommended that the latest version of NXP's free GUI Guider tool also be downloaded

Download this from:
<https://www.nxp.com/GUI-GUIDER>

7.4 NXP MCUBootUtility

This optional tool supports serial download modes (USB-HID and UART) as well as various utilities for loading flash memory

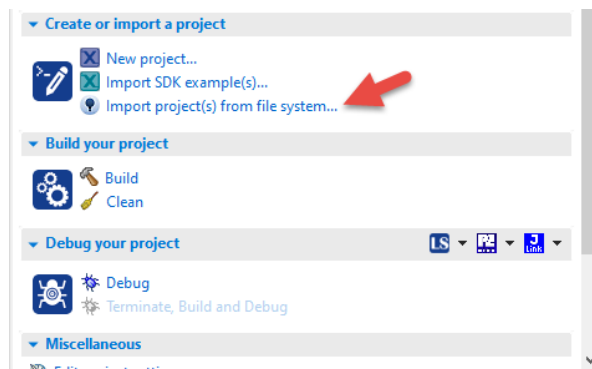
Download this from:
<https://github.com/JayHeng/NXP-MCUBootUtility>

8 Development Environment

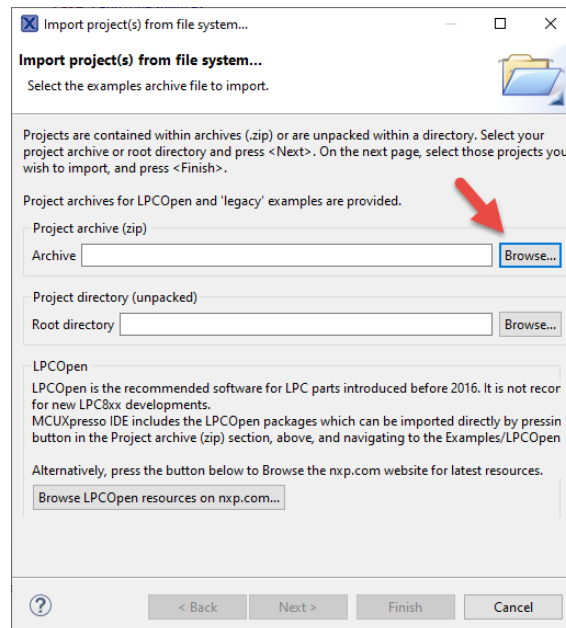
Follow the instructions from NXP to install the MCUXpresso IDE and SDK, the documentation to assist with setting up the development environment are provided in section 0.

8.1 Importing a Project zip File

Once the MCUXpresso IDE and SDK are installed, import the project zip file into a new workspace by clicking on the “Import project(s) from file system link” on the Quickstart Panel.

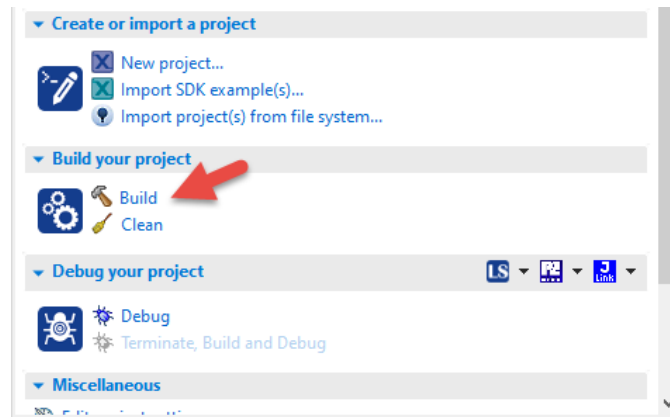


Select the zip file to import from the file system by pressing the browse button and then press finish.



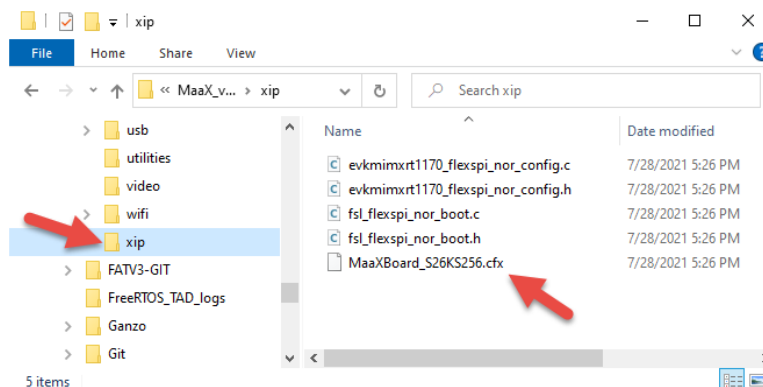
8.2 Building projects

To build the selected project, simply press the **Build** button from the **Quickstart** panel.

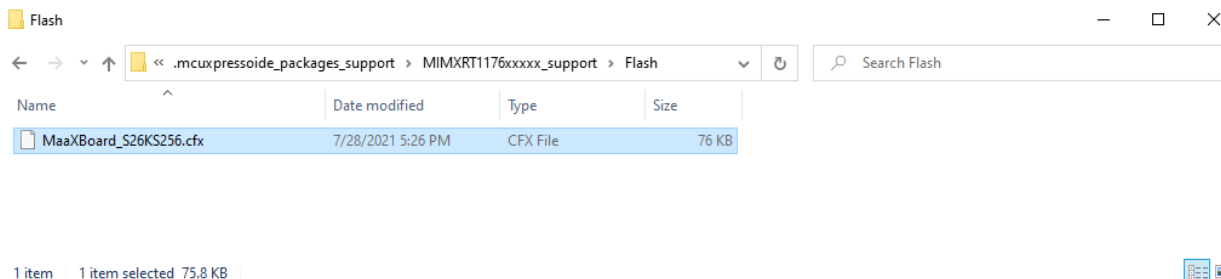


8.3 Setting up the debugger

Before attempting to debug or flash the board the `MaaXBoard_S26KS256.cfx` file has to be added to the `.mcuxpressoide_packages_support` folder. To do this, locate the `MaaXBoard_S26KS256.cfx` file in the project directory `xip` folder.

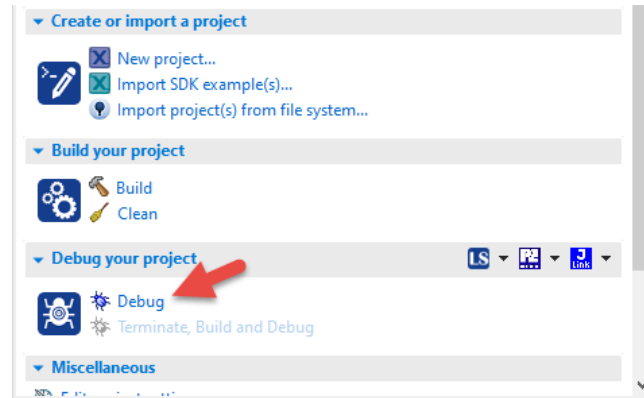


Copy the `MaaXBoard_S26KS256.cfx` file and place it in the `.mcuxpressoide_packages_support\MIMXRT1176xxxxxx_support\Flash` located within the project workspace directory.



8.4 Downloading and running the application

At this point, the application is ready for debugging. Press the `Debug` button from the `Quickstart` Panel



Once the application is downloaded, press the `resume` button to run the application.



Once the application is downloaded and running, press the `terminate` button to exit debugging.



Power-cycling the board or pressing the reset button will restart the application since it is now located in flash.

9 Porting NXP RT1170-EVK SDK examples to MaaXBoard RT

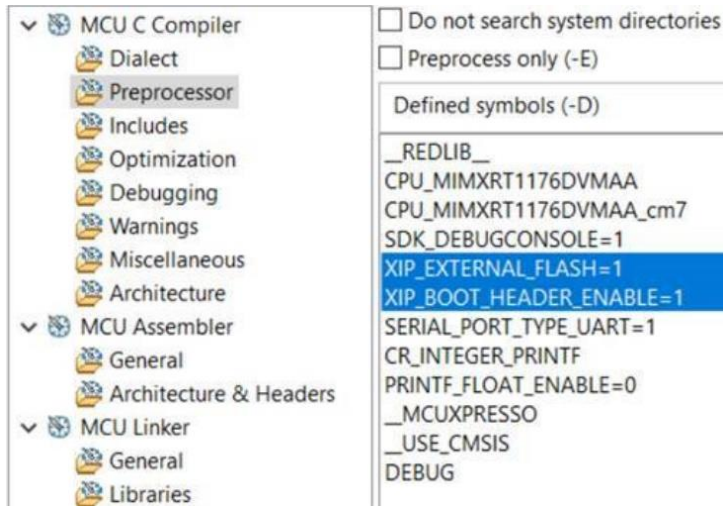
To run NXP SDK examples on MaaXBoard RT, minor changes are typically required to the example project and its source code, to make these compatible with the mapping and devices on MaaXBoard RT hardware. These changes are summarised in the table below.

Note! For each of these cases use the applicable code in *Ref.Design #1* for guidance

#	Item to change	Detail of changes to be implemented
1	Flash memory configuration	Change Project Settings → C/C++ Build → MCU Settings, to define the Flash memory on MaaXBoard RT: a) Flash size = 0x20000000 (32 MB of Hyperflash) b) Flash driver = MaaXBoard_S26KS256.cfx c) Default LinkServer Flash driver = MaaXBoard_S26KS256.cfx
2	SDRAM configuration	Change Project Settings → C/C++ Build → MCU Settings, to define the SDRAM memory on MaaXBoard RT: a) SDRAM size = 0x20000000 (32 MB) b) Replace the dcd.c file in the imported project's board folder to change the SDRAM interface width to x16 (not x32 width as used on the EVK)
3	USER Button	Same pin T8 is used (GPIO_WAKEUP) - <i>No change is needed</i>
4	USER LED	MaaXBoard RT provides an RGB User LED, where Red, Green and Blue LEDs are driven by GPIO pins R15, N3, R17 . Remap to one of these pins.
5	UART interfaces	LPUART1 (Debug Console) no change. Pinned-out to 18- and 40-pin headers LPUART2 (Unassigned). Pinned-out to 40-pin headers LPUART10 (Bluetooth HCI interface).
6	I2C interfaces	I2C2, I2C3, I2C5 and I2C6 are pinned-out. When porting SDK examples that utilize I2C interfaces, the I2C interfaces need to be reviewed and modified as necessary depending upon usage.
7	USB interfaces	Change the USB Host controller from USB0 to USB1 NXP SDK USB host examples specify kUSB_ControllerEhci0 , while for MaaXBoard RT this must be changed to kUSB_ControllerEhci1 (in api.h)
8	Ethernet interfaces	100M PHY : Microchip KSZ8081RNBCA is same device as used on the EVK 1G PHY : Microchip KSZ9131RNX is used on MaaXBoard RT (this replaces the Realtek RTL8211FDI PHY used on RT1170-EVK)
9	Codec interface	NXP SGTL5000 Codec (replaces Wolfson WM8960), connected via SAI2 (not SAI1). This means that NXP SDK example audio applications require modification before they can be run on MaaXBoard RT
10	Touchscreen controller driver	Different touchscreen controller, use fsl_ft5406_rt.c driver. Change I2C controller interface from I2C5 to I2C6 . Remap the DSI_TS_nINT interrupt pin.
11	MIPI Display controller driver	Different display and MIPI controller, use Avnet-supplied driver (avt-ili9881c.c) plus related modifications. MIPI-DSI signals have same mapping as on EVK. DSI_EN and DSI_BL_PWM control signals require remapping.
12	Wi-Fi interface	Pin N14 drives WLREG_ON WIFI_WAKE signal not supported
13	Bluetooth interface	LPUART10 replaces LPUART2 for Bluetooth HCI UART interface BT_WAKE signal not supported

9.1 Building and running SDK RT1170-EVK Examples

- 1) In the **Project Settings → C/C++ Build → Settings → Preprocessor** setting:
When XIP flash options = 0, the code executes in on-chip RAM
When XIP flash options = 1, the code executes-in-place on HyperFlash (see screenshot below)



- 2) Make sure the MaaXBoard_S26KS256.cfx flash driver is specified in the memory settings, next to the flash entry. This needs to be explicitly specified there (as well as in default Flash Driver field). This file needs to be in same workspace.
- 3) If a flash-related error is reported after launching Debug, execute a manual Mass Erase (or launch Debug a second time) to correct this. The debug session will then launch and execute application correctly
- 4) Accommodating the subset of hardware differences between MaaXBoard RT and the RT1170-EVK, a broad range of RT1170-EVK SDK examples have already been adapted and tested on MaaXBoard RT. This material will be made available in due course via public Avnet Github pages

10 MaaXBoard RT Example Applications

10.1 Custom System-level Reference Designs

#	Application Name	Board functions exercised	Repo / Location
1	<i>Ref.Design #1:</i> Out of Box Demo	MIPI DSI display, touchscreen, USB host, Wi-Fi scan and connect, 100M & 1G Ethernet, PDM microphones, audio output, RGB LED, CLI interface, HyperFlash, User button, CSI camera interface, I2C interface.	<i>Not yet posted to public repo</i>
2	<i>Ref.Design #2:</i> Wi-Fi Webserver	Wi-Fi client, Wi-Fi softAP, dual-core inter-core communication, headless webserver, websockets, I2C sensors, Pi HAT interface, data charting, HyperFlash, runtime storage, lwIP, FXOS8700 IMU motion measurements, VL53L3CX Time-of-Flight measurements	"
3	<i>Ref.Design #3:</i> BLE Shell plus Mobile UI	BLE/Smartphone interface, RGB LEDs, User button, console interface. More details to follow	"

10.2 Adaptations of NXP RT1170-EVK SDK Examples

Tabled below is a sampling of SDK examples that have been adapted to run on MaaXBoard RT
(These and other ported examples will be available soon after release of the hardware via Github)

#	Application Name	Board functions exercised	Repo / Location
4	gpio_test_cm7	RGB LEDs, User button, console interface	<i>Not yet posted to public repo</i>
5	igpio_input_interrupt_cm7	GPIO interrupt input example	"
6	host_hid_mouse_bm_cm7	USB host mouse device	"
7	host_msd_fatfs_bm_cm7	USB host mass storage device (thumb-drive) test	"
8	dev_cdc_vcom_bm_cm7	USB host CDC virtual COM port	"
9	dev_cdc_vcom_freertos_cm7_passthrough	USB passthrough bridge application to send AT commands to cellular modem	"
10	mbedtls_benchmark_cm7	Memory benchmarking application	"
11	wifi_iperf_cm7	Wi-Fi iPerf throughput benchmarking test of the Murata 1ZM Wi-Fi/BT module	"
12	tensorflow_lite_micro_label_image_cm7		
13	littlevgl_demo_widgets_bm_cm7	LVGL graphics MIPI 7" display test	"
14	littlevgl_guider_cm7	LVGL GUI Guider MIPI 7" touch display test	"
15	csi_mipi_rgb_cm7	CSI MIPI camera format conversion and video output on MIPI-DSI 7" display	"
16	pdm_sai_edma_cm7	PDM mics and Codec I/O audio test	"
17	lwip_httpsrv_bm_cm7	LWIP Ethernet based HTTP webserver	"
18	driver_xip_board_MIMXRT1176_cm7	Hyperflash flashloader driver generation and test application	"

11 Customizing the OOB Test Suite (Ref. Design #1)

11.1 Overview of MaaXBoard-RT Reference Design #1

This reference design is the “out-of-box” application that is factory pre-programmed into the HyperFlash memory of new MaaXBoard RT boards. It provides a suite of board test functions that can be exercised using two different interfaces:

- Command-line (serial console) interface or
- Graphical UI (using touch or a mouse pointer)

Developers are encouraged to build and customize this application, using the following tools and enablement from NXP:

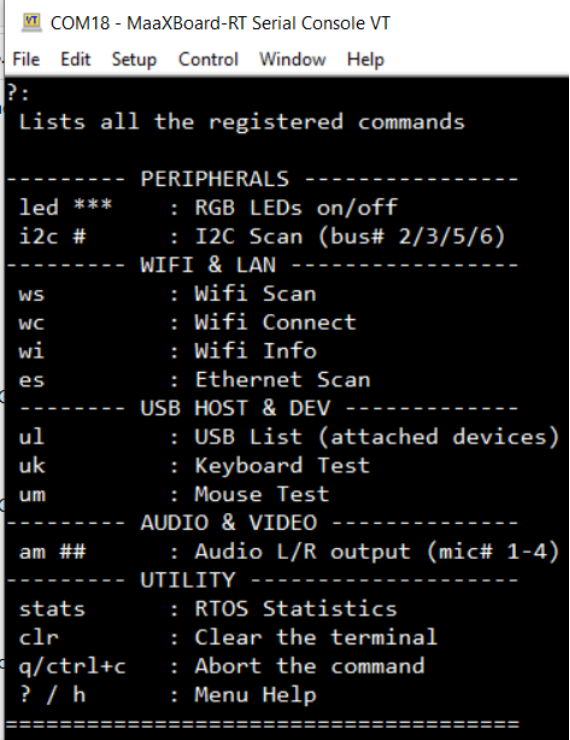
- NXP MCUXpresso IDE
- NXP GUI Guider LVGL graphic editor & code-generation tool
- NXP SDK for RT1170-EVK

The application is partitioned into different FreeRTOS tasks that can be enabled or disabled as needed. This facilitates rapid customization and extensions to this embedded application

The **CLI menu** interface can be used to exercise all board functions, it can also be extended by developers to add new shortcut commands.

Functions exercised via the CLI Menu shortcuts include the following:

- **GPIO:** Keyboard control of RGB LEDs and reporting of User button events
- **I2C:** Scans & reports I2C devices onboard. or connected via the two expansion headers
- **Wi-Fi:** Wi-Fi Scan, Wi-Fi Connect and Wi-Fi Info WLAN reporting functions
- **Network:** Reports IP addresses for any connected Ethernet ports
- **USB Host:** Reports the presence of keyboard and mouse HID devices
- **AV Audio:** Routes PDM mics via Codec to the stereo audio jack.
- **AV Video:** Reports if MIPI-CSI Camera is present
- **Custom:** Supports Menu extension for custom expansion hardware
- **Statistics:** Reports utilization of Cortex-M7 processor by the runtime RTOS tasks



```

COM18 - MaaXBoard-RT Serial Console VT
File Edit Setup Control Window Help
?:
Lists all the registered commands

----- PERIPHERALS -----
led *** : RGB LEDs on/off
i2c #   : I2C Scan (bus# 2/3/5/6)
----- WIFI & LAN -----
ws      : Wifi Scan
wc      : Wifi Connect
wi      : Wifi Info
es      : Ethernet Scan
----- USB HOST & DEV -----
ul      : USB List (attached devices)
uk      : Keyboard Test
um      : Mouse Test
----- AUDIO & VIDEO -----
am ##   : Audio L/R output (mic# 1-4)
----- UTILITY -----
stats   : RTOS Statistics
clr     : Clear the terminal
q/ctrl+c : Abort the command
? / h   : Menu Help
=====
  
```

The **GUI display** interface (requires optional MIPI LCD display), provides an alternative method to control all the functions listed for the CLI user interface, as well as view feedback via a rich graphical UI, activating these functions either via touch interface, or using a mouse

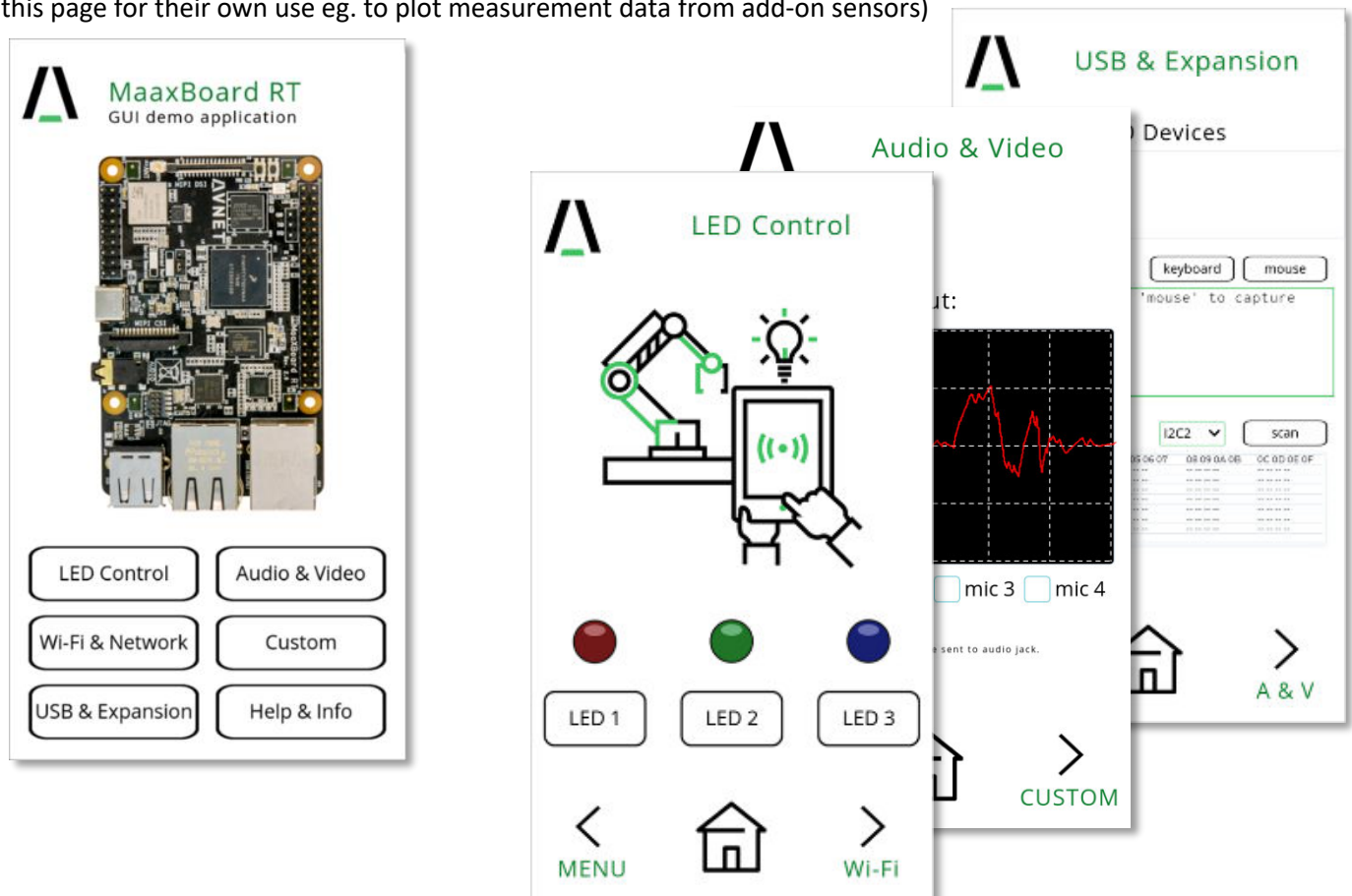
The project source files are available for users to download, edit and repurpose for accelerated development of customized user applications

(Avnet's MaaXBoard RT "Factory Test" application is an example of how this codebase can be rapidly repurposed, in that case to interface with an automated C# test application running on a Windows computer)

The Graphical UI screens are designed using NXP's free GUI Guider tool. As shipped, this application supports 7 different screens which can be customized or added to:

- 1) Shortcut Menu
- 2) LED Control
- 3) Wi-fi & LAN Networking
- 4) USB & I2C Expansion
- 5) Audio & Video
- 6) Custom (User Sandbox)
- 7) Help & Info

The **Custom** GUI page by default charts the Wi-Fi signal strength, but users are encouraged to repurpose this page for their own use eg. to plot measurement data from add-on sensors)



11.2 Console mode (CLI / Command-Line Interface)

Users without the optional MIPI 7-inch display can run the application entirely from the serial console interface. MaaXBoard RT's CLI is configured to use LPUART1 configured with a baudrate of 115200 8N1 (accessible via the **J17** Debug UART 3-pin header).

*Note: If no LCD GUI display is in use, set the relevant **#define** in **globals.h** to zero ie.*

```
#define GUI_EN    0

18 /*****
19  * Definitions
20  *****/
21 /* Freertos task can be enabled by setting 1, disabled by setting 0 */
22 #define GUI_EN    1
23 #define WIFI_EN    1
24 #define ETH100MB_EN 1
25 #define ETH1GB_EN  1
26 #define USB_PERIPH_EN 1
27 #define CONSOLE_EN 1
28 #define AUDIO_EN   1
```

Figure 13 – Configure GUI_EN in global.h

```
----- PERIPHERALS -----
led ***      : RGB LEDs on/off
i2c #        : I2C Scan (bus# 2/3/5/6)
----- WIFI & LAN -----
ws           : Wifi Scan
wc           : Wifi Connect
wi           : Wifi Info
es           : Ethernet Scan
----- USB HOST & DEV -----
ul           : USB List (attached devices)
uk           : Keyboard Test
um           : Mouse Test
----- AUDIO & VIDEO -----
am ##       : Audio L/R output (mic# 1-4)
----- UTILITY -----
stats        : RTOS Statistics
clr          : Clear the terminal
q/ctrl+c    : Abort the command
? / h       : Menu Help
===== Avnet GUI Demo v1.0 [2021-06-25]
```

Figure 14 – Default list of supported CLI Menu commands

11.3 GUI mode (Graphical UI)

To run in GUI mode, the optional 7inch MIPI display (720x1280) from Avnet is required.
The Menu can be navigated using the touchscreen or mouse (attach via the USB type-A host connector).

Depending on the required orientation, the user may want to rotate the screen by 180 degree.
To do this, **AVT_DISPLAY_ROTATE_180** must be defined in the preprocessor under project settings.

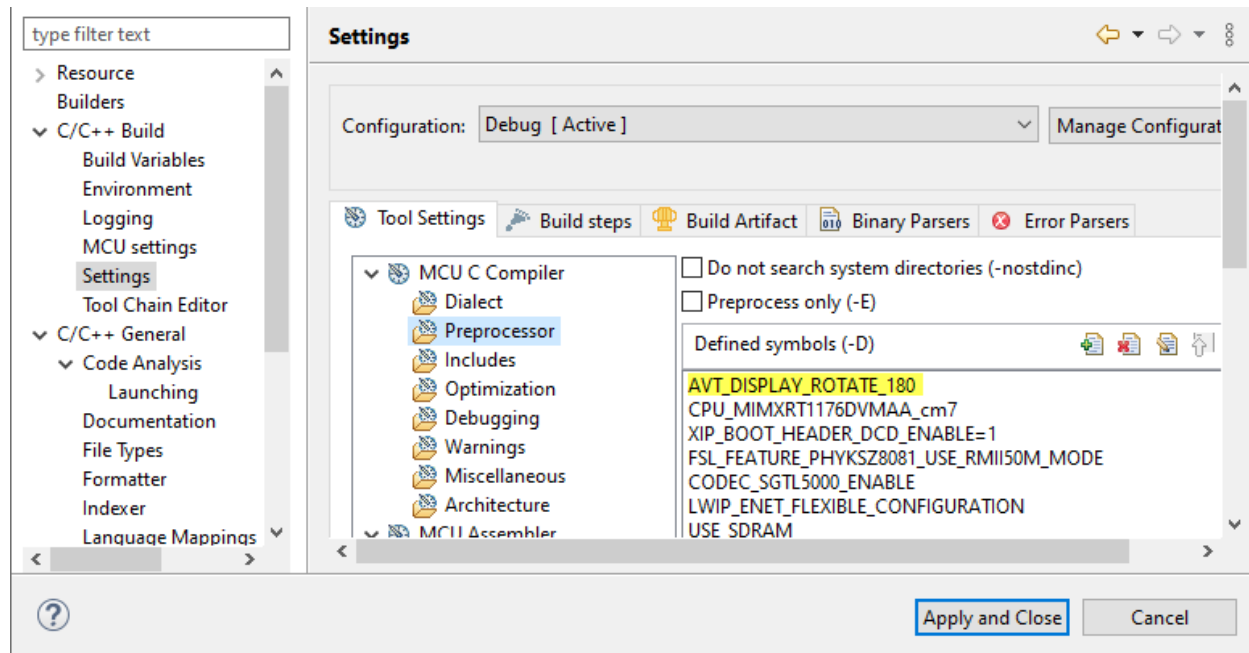
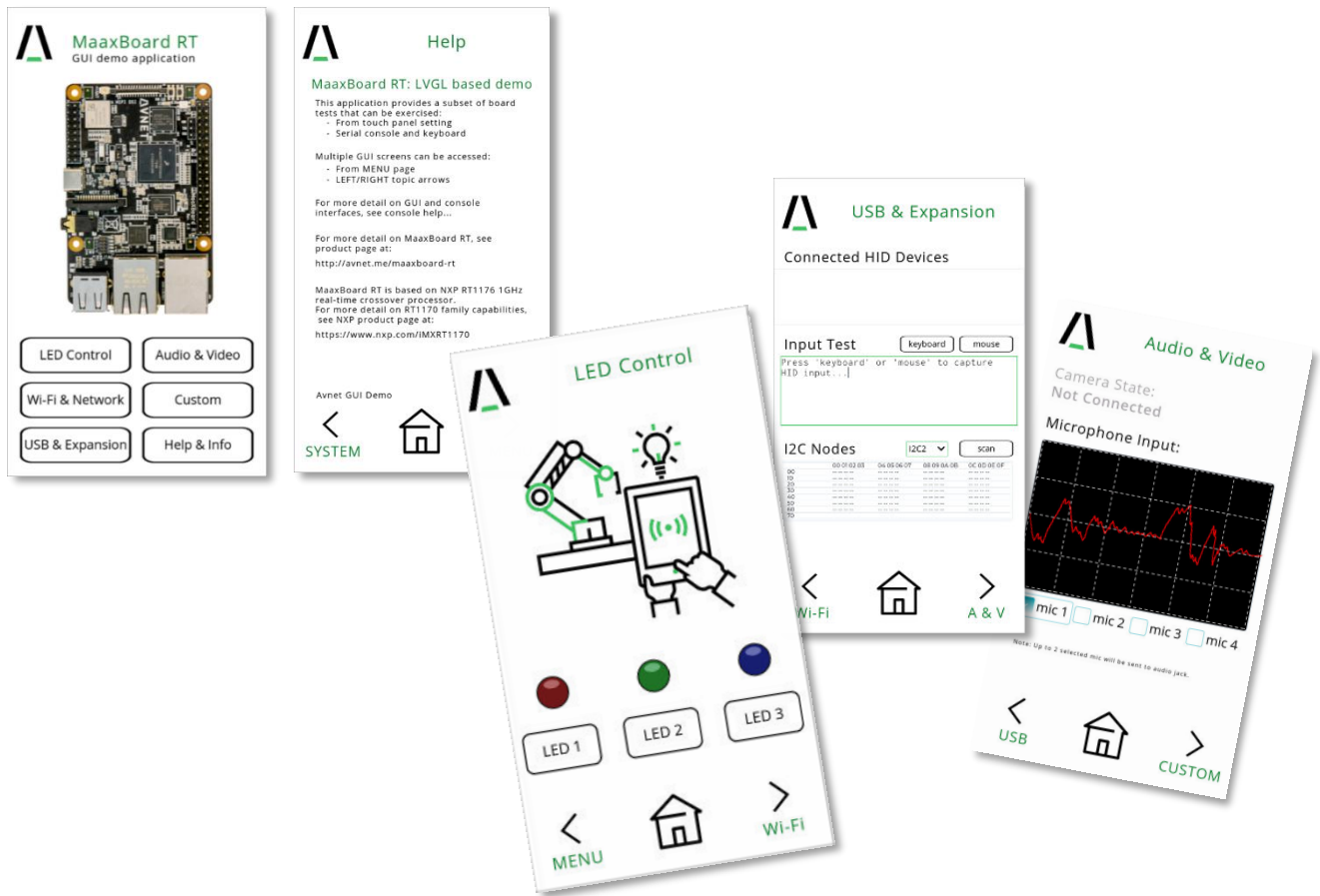


Figure 15 – Screen orientation in project settings



11.4 Project Structure for Reference Design #1

The project contains several folders however, only a few are of interest for the purpose of this document. The folders of interest are the following:

- **generated/**
 - GUI Guider generated source files
- **sources/**
 - application source files
- **board/**
 - **board.c/h**
 - board specific definitions
 - **clock_config.c/h**
 - generated clock configuration from the MCUXpresso configurator
 - **pin_mux.c/h**
 - generated pin configuration from the MCUXpresso configurator
- **littlevgl/**
 - LVGL graphics library version 7.11
- **wifi/**
 - Wi-Fi driver and Bluetooth firmware

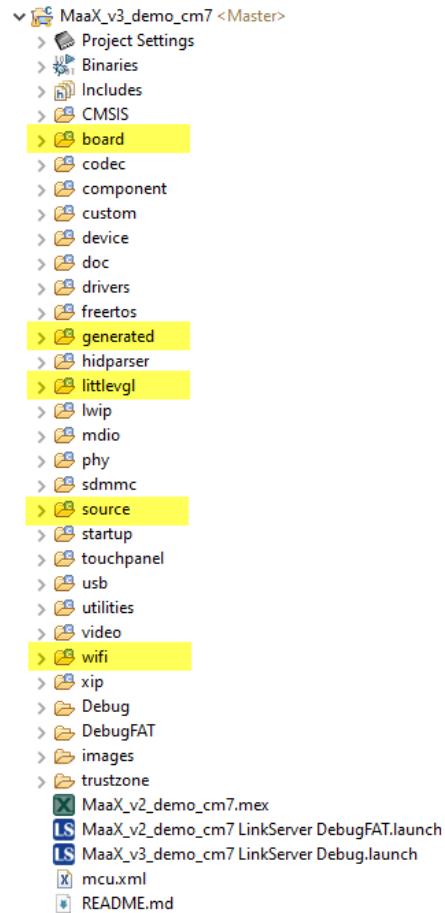


Figure 16 – Project folders

The main function `int main(void)` is located under `source/littlevgl_guider.c` and contains the initialization routines as well as the logic for creating the required FreeRTOS tasks.

There are a total 10 FreeRTOS tasks running.

- **lvgl_task**
 - Task handling all GUI object events and navigation
- **wifi_task**
 - Task obtains Wi-Fi network information, performs SSID scanning, and connects to a hardcoded Wi-Fi network. The SSID can be changed from the default in `network_demo.c` by changing the following defines


```
#define EXT_AP_SSID          "max123"
#define EXT_AP_PASSPHRASE   "1qaz2wsx"
```
- **eth_100m_task**
 - 100Mb DHCP client to obtain an IP address from the network
- **eth_1g_task**
 - 1Gb DHCP client to obtain an IP address from the network

- **USB_HostTask**
 - USB host task for enumerating HID devices (keyboard and mouse)
- **USB_HostApplicationMouseTask**
 - Reads HID mouse data
- **USB_HostApplicationKeyboardTask**
 - Reads HID keyboard data
- **USB_logTask**
 - Responsible for processing the USB keyboard and mouse data and outputs to the serial port. This task facilitates the command line user interface (CLI).
- **console_task**
 - Processes console commands received over the serial port

11.5 Adding a new CLI Command

MaaXBoard-demo uses the standard [FREERTOS+CLI](#) framework. Adding new CLI commands can be achieved with the following 3 steps. Refer to existing CLI command implementation examples located in `source/UART_CLI.c`

1. Create the new function with the parameters shown below and place it below the existing `ethernetScanCommand` function body. The `pcWriteBuffer` parameter represents the output buffer that would be sent out to the serial port console after command execution.

```
static BaseType_t newFunction( char *pcWriteBuffer, size_t,
                             xWriteBufferLen, const char *pcCommandString )
{
    /* your logic here */
};
```

2. Create the new command structure following the format shown below and place it after the `ethernetScanCommandStruct` structure definition. (Commands may contain one or more parameters)

```
static const CLI_Command_Definition_t taskNewFuncCommandStruct =
{
    "newfunc",
    "----- UTILITY -----\r\n"
    " newfunc      : newfunc description \r\n",
    newFunction,
    0
};
```

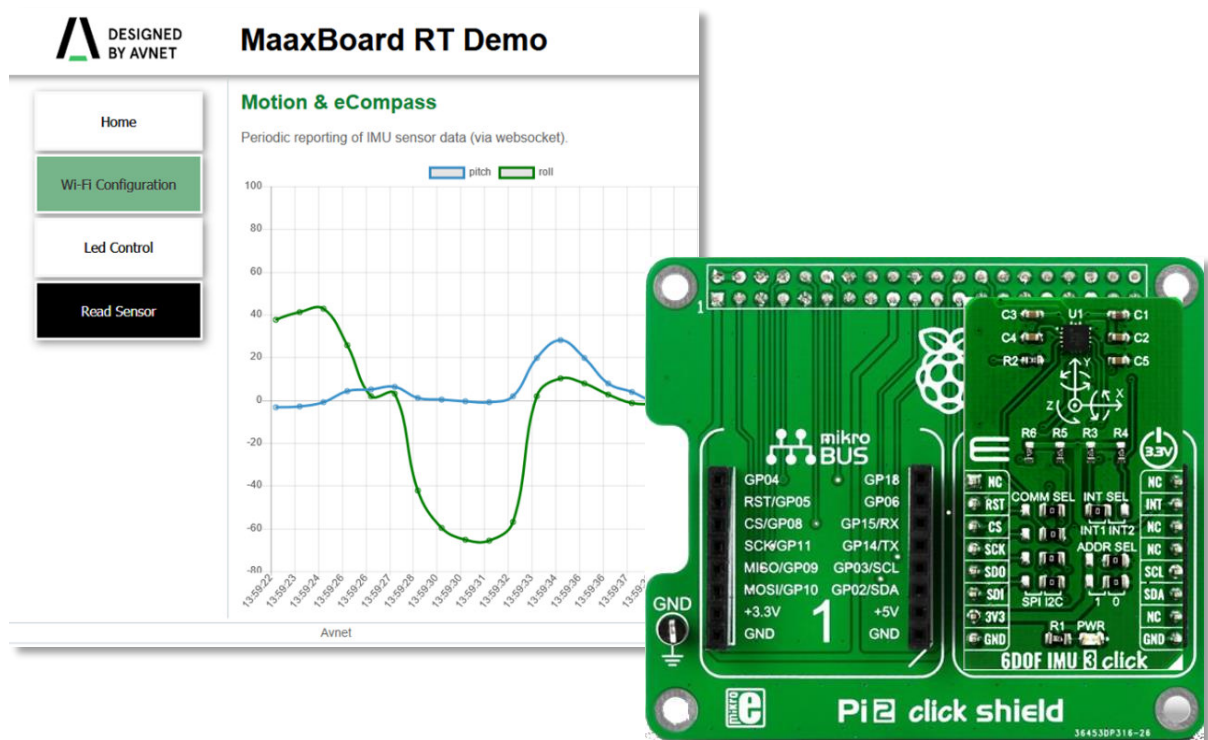
3. For the new command to be recognized, it must be registered. Register the newly created command as follows and place it in the `console_task` function along with all the other command registrations.

```
FreeRTOS_CLIRegisterCommand(&taskNewFuncCommandStruct);
```

12 Customizing the Wi-Fi Webserver (Ref. Design #2)

12.1 Overview of the MaaXBoard-RT Reference Design #2

- FreeRTOS based dual-core application supports
 - M7 based webserver and Wi-Fi network connectivity
 - M4 based I2C sensor monitoring & output control
 - Remote, headless operation
 - UI access to board via any internet browser
 - Websocket streaming of 6-axis IMU sensor data
 - Network connection via 802.11ac Wi-Fi
 - BLE-connected Smartphone App support later
 - Charts accelerometer and magnetometer data from FXOS8700 IMU sensor (NXP)
 - Charts distance measurements from VL53L3CX Time-of-Flight sensor (ST)
- (For the charting of sensor measurements the following optional hardware needs to be added)
- [PI 2 Click Shield](#) / HAT (MikroE, \$8.00)
 - [6DOF IMU 3 Click](#) board (MikroE, \$7.00)
 - [LightRanger 8 Click](#) board (MikroE, \$12.00)



This application utilizes the M7 and the M4 cores to demonstrate a headless webserver on the i.MXRT1170 MCU. Main features of this design include the following:

- Utilizing the HyperFlash for non-volatile configuration storage
- Wi-Fi softAP or Wi-Fi client
- HTTP server based on lwIP stack
- Inter-core communication between the M7 and M4 cores.
- IMU, distance sensor reading using i2c bus.

12.2 Modes of Operation

12.2.1 HyperFlash Partitioning

The onboard HyperFlash memory size is 32MB. In this application the flash memory is partitioned into two equal sections of 16MB. The first 16MB section is used as program flash and contains the running application. The subsequent 16MB section is used for storage.

(Note: The flash partitioning can be changed in the project settings.)

The following definition must be set in the project. This value represents the starting physical address of the configuration partition. (16777216 = 0x1000000)

MFLASH_FILE_BASEADDR=16777216

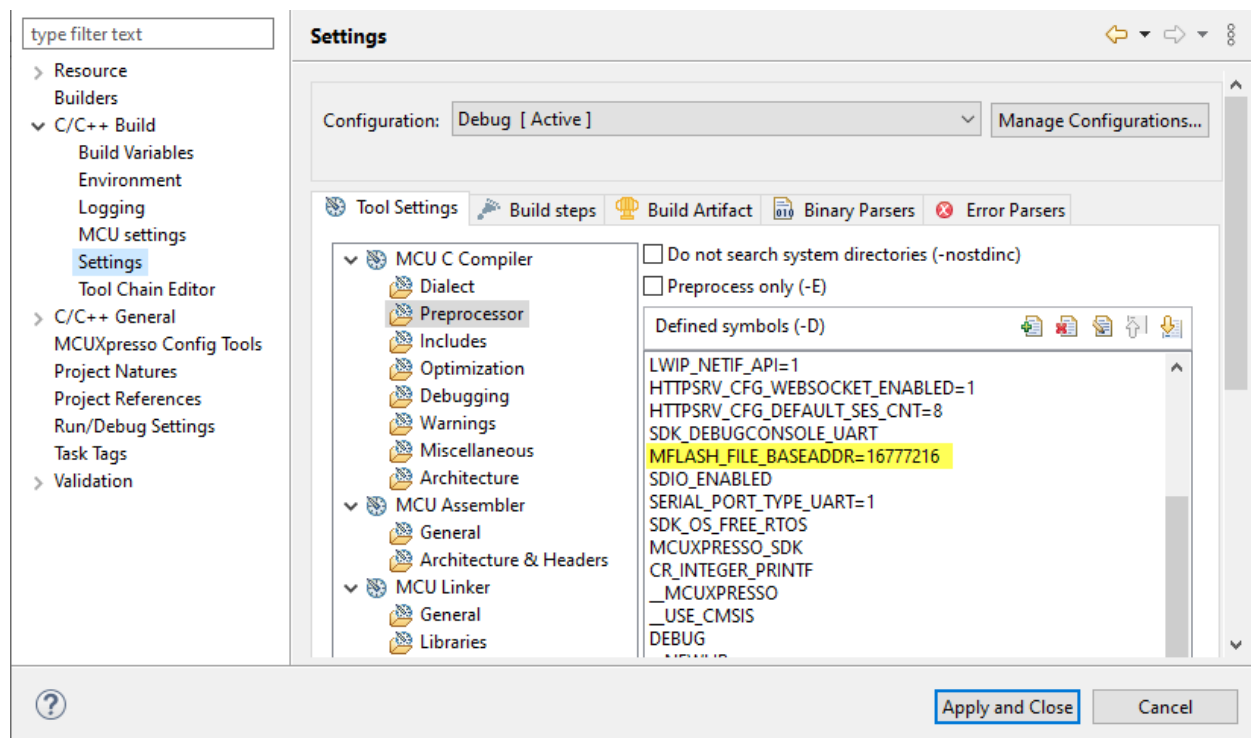


Figure 17 – HyperFlash Base address

12.2.2 Soft AP mode

MaaXBoard RT will run in soft AP mode if SSID and password have not been stored in the HyperFlash configuration partition. Alternatively, the user can pre-configure the board's Wi-Fi network connection, by hard-coding the default SSID and password in `webconfig.h`

```
#define WIFI_SSID "maaxboard_access_point"
#define WIFI_PASSWORD "maaxboard123"
```

Once a connection is established, use the connected device browser to browse to the boards IP address: <http://192.168.1.1> This will open the boards webpage through which an SSID and password can be entered and stored. The board can now be operated in client mode.

12.2.3 Wi-Fi Client Mode

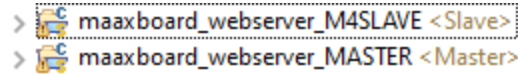
If a valid SSID and password have been stored in HyperFlash, then MaaXBoard RT will enter client mode. In client mode, MaaXBoard RT will connect to the specified network using the available SSID and password. Connecting a serial console to the board's serial port allows the user to view additional details pertaining to this connection. (*Note: Serial terminal configuration is 115200/8-N-1*)

```
Starting MaaXBoard Webserver DEMO
[i] Trying to load data from mflash.
[i] Saved SSID: SSID, Password: password
[i] Initializing WiFi connection...
MAC Address: XX:XX:XX:XX:XX:XX
[net] Initialized TCP/IP networking stack
WLAN initialized
WLAN FW Version: w8987o-V0, RF878X, FP91, 16.91.10.p200, WPA2_CVE_FIX 1,
PVE_FIX 1
[i] Successfully initialized WiFi module
Connecting as client to ssid: ssid with password password
    Connected to following BSS:SSID = [SSID], IP = [192.168.0.25]
[i] Connected to Wi-Fi
ssid: SSID
[!]passphrase: password
    Now join that network on your device and connect to this IP: 192.168.0.25
```

To view the web page output by the MaaXBoard RT webserver, the user should navigate their internet browser to the IP address shown in the serial terminal.

12.3 Project Structure for Reference Design #2

This reference design application is partitioned into two projects, one for each core, with each project running a set of FreeRTOS tasks.



The M7 core project (MASTER) runs FreeRTOS with memory scheme 3 and runs 3 tasks.

- **wifi_task**
 - This task initializes Wi-Fi connectivity
- **http_srv_task**
 - HTTP server task based on lwIP stack
- **app_task**
 - Communication task responsible for retrieving sensor data from the M4 core

The M4 core project (SLAVE) runs FreeRTOS with memory scheme 4 and runs 3 tasks.

- **IMU_TASK**
 - polls the IMU sensor for data every 10ms (refer to [6DOF IMU 3 CLICK](#))
- **LR_TASK**
 - Polls the LightRanger sensor every 500ms (read [LightRanger 8 CLICK](#))
- **MC_TASK**
 - sends sensor data to M7 core every 200ms across shared memory

Inter-core communication uses a [message buffer](#) with statically allocated memory **rpmsg_sh_mem** at location 0x202c0000 with size of 0x2000. This shared memory is at a fixed location and accessed by both cores. This shared memory can be adjusted in the project properties under the MCU settings section.

Type	Name	Alias	Location	Size	Driver
Flash	BOARD_FLASH	Flash	0x30000000	0x1000000	MaaXBoard_S26KS256.cfx
RAM	BOARD_SDRAM	RAM	0x80000000	0x2000000	
RAM	OCRAM_ITCM_ALIAS	RAM2	0x20200000	0x20000	
RAM	SRAM_DTC_cm7	RAM3	0x20000000	0x40000	
RAM	SRAM_OC1	RAM4	0x20240000	0x80000	
RAM	rpmsg_sh_mem	RAM5	0x202c0000	0x2000	
RAM	NCACHE_REGION	RAM6	0x20300000	0x40000	

12.3.1 M7 (MASTER) Wi-Fi Webserver project

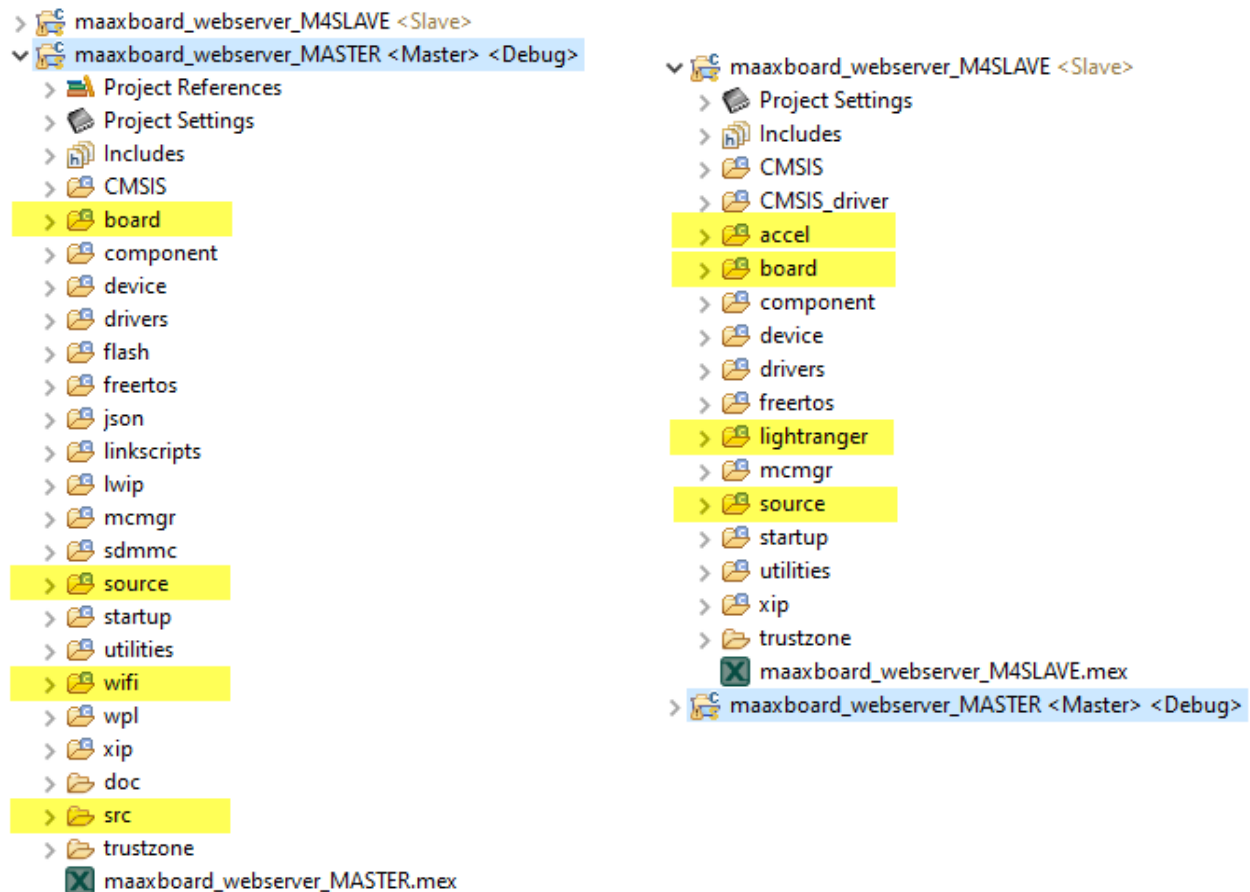
This contains several folders, but only the following are of interest for purpose of this document.

- **src/**
 - web source files (.css .js .html)
- **source/**
 - application source files
- **board/**
 - **board.c/h**
 - board specific definitions
 - **clock_config.c/h**
 - generated clock configuration from the MCUXpresso configurator
 - **pin_mux.c/h**
 - generated pin configuration from the MCUXpresso configurator
 - lvgl graphics library version 7.11
- **wifi/**
 - Wi-Fi driver and Bluetooth firmware

12.3.2 M4 (SLAVE) Sensor Service project

The main folders are the following:

- **accel/**
 - IMU driver
- **source/**
 - application source files
- **board/**
 - **board.c/h**
 - board specific definitions
 - **clock_config.c/h**
 - generated clock configuration from the MCUXpresso configurator
 - **pin_mux.c/h**
 - generated pin configuration from the MCUXpresso configurator
 - lvgl graphics library version 7.11
- **lightranger/**
 - LightRanger driver



12.4 Installing the Click Boards

The **LightRanger Click** sensor board must be installed in slot #1 on the click shield, and the **IMU Click** sensor board must be installed in **slot #2**. It is not possible to interchange the board locations due to physical pin routing.

12.5 Running the demo

Connect a serial terminal to the MaaXBoard RT serial port using the MCU-LINK serial port feature to view the log. Observe the IP address and use your browser to navigate to that address to view the MaaXBoard RT web page. (Note: Serial terminal configuration is 115200/8-N-1)

The log shown below is a sample output of the **maaxboard_webserver_MASTER** demo as displayed in the terminal window:

```
Starting MaaXBoard Webserver DEMO
[i] Trying to load data from mflash.
[i] Saved SSID: ssid, Password: password
[i] Initializing WiFi connection...d
MAC Address: XX:XX:XX:XX:XX:XX
[net] Initialized TCP/IP networking stack
WLAN initialized
WLAN FW Version: w8987o-V0, RF878X, FP91, 16.91.10.p200, WPA2_CVE_FIX 1,
PVE_FIX 1
[i] Successfully initialized WiFi module
Connecting as client to ssid: ssid with password password
    Connected to following BSS:SSID = [ssid], IP = [192.168.0.13]
[i] Connected to Wi-Fi
ssid: ssid
[!]passphrase: password
Now join that network on your device and connect to this IP: 192.168.0.13
```

12.6 HyperFlash as runtime storage

SDK 2.10 provides a standard software component "**mflash rt1170**" based on NOR flash memory. Since MaaXBoard RT is instead fitted with 32MB HyperFlash. Some changes are required to be made to the mflash driver code. The affected files are **mflash_drv.c** and **mflash_drv.h**

- **flash/**
 - **mimxtr1170/**
 - **mflash_drv.c** (*modified*)
 - **mflash_drv.c** (*modified*)
 - **mflash_common.h**
 - **mflash_file.c**
 - **mflash_file.h**

The **mflash_file.c** file provides a simple statically allocated flat filesystem used with FLASH memories. It is critical to note that when using the HyperFlash for storage in a multi core project, the user must disable the flexspi1 clock configuration on the slave core, in our case, the M4 core. This can be achieved by qualifying the flexspi1 clock configuration with the **FLEXSPI_IN_USE** definition. Disabling the slave Flexspi1 clock on the slave core is performed in **clock_config.c**

Using **FLEXSPI_IN_USE** definition in the slave project.

```
#if !(defined(XIP_EXTERNAL_FLASH) && (XIP_EXTERNAL_FLASH == 1) ||
defined(FLEXSPI_IN_USE))
    rootCfg.mux = kCLOCK_FLEXSPI1_ClockRoot_MuxOscRc48MDiv2;
    rootCfg.div = 1;
    CLOCK_SetRootClock(kCLOCK_Root_Flexspi1, &rootCfg);
#endif
```

12.7 User Customization

12.7.1 Frontend (Webserver UI)

All front-end web resources can be found in `src/wifi_common/webconfig/webui`

The LWIP stack provides a PERL script `mkfs.pl` which is used to convert entire web frontend sources into a single `.c` file. The generated `.c` file can then be added to a project and compiled. The `mkfs.pl` script is located here, `src/wifi_common/webconfig/webui/mkfs.pl`

The `mkfs.pl` script requires only the folder location of the web resources as a parameter. The script converts all `.jpg`, `.html`, `.css`, `.js`, ... into `httpsrv_fs_data.c` in the form of constant arrays and their pointers.

The `mkfs.pl` script is invoked in the following manner.

```
perl mkfs.pl webui
```

The current project utilizes <8% of the 32 MB flash, leaving the user substantial space for customization.

12.7.2 Backend (LWIP, CGI, HTTP)

LWIP webserver implementation supports CGI (Common Gateway Interface). The webserver receives and processes http GET/POST requests and invokes the appropriate registered call back functions.

HTTP callback functions are defined in `source/main_m7.c` in a CGI link table `cgi_lnk_tbl[]`
The user may define additional callback functions as needed or modify the existing ones.

eg. GET 192.168.xx.yy/led.cgi will invoke `cgi_led()` function.

```
const HTTPSrv_CGI_LINK_STRUCT cgi_lnk_tbl[] = {
    {"led", cgi_led},           // led control
    {"reset", CGI_HandleReset}, // reset wifi
    {"get", CGI_HandleGet},     // wifi scan
    {"imu", CGI_HandleGetSensor}, // get IMU, lightranger sensor value
    {"post", CGI_HandlePost},   // set new SSID, password
    {"status", CGI_HandleStatus}, // get wifi status
    {0, 0} // DO NOT REMOVE - last item - end of table
};
```

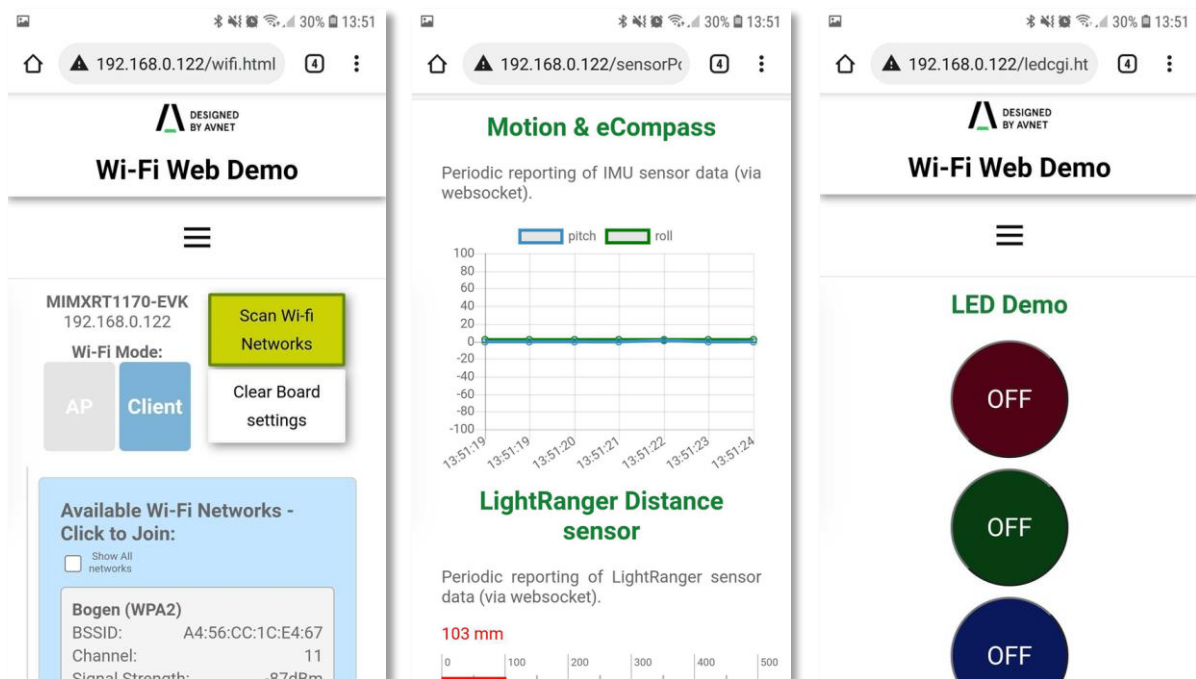
Note: Both GET / POST methods will invoke the given callback function.

Example of a callback function:

```
static int CGI_Example(HTTPSrv_CGI_REQ_STRUCT *param)
{
    char[] data = "OK"; // dummy text data
    HTTPSrv_CGI_RES_STRUCT response = {0};
    response.ses_handle      = param->ses_handle;
    response.status_code     = HTTPSrv_CODE_OK;
    response.content_type    = HTTPSrv_CONTENT_TYPE_PLAIN;
    response.data            = data;
    response.data_length     = strlen(data);
    response.content_length  = response.data_length;
    HTTPSrv_cgi_write(&response);
    return (response.length);
}
```

12.7.3 Webserver access via Smartphone

Access to the Webserver pages via smartphone Wi-Fi network connection is easily accommodated. Shown here are webpage examples reformatted for viewing and control from the smaller screen



13 Known Issues

The following should be noted when working with the Rev.3 version production MaaXBoard RT

#	Description of Issue	Comment / Workaround
1	When MCU-LINK ribbon cable is attached, the application will be held in reset state if debugger is unpowered (ie. it's USB cable is disconnected from the development PC)	<u>Debugger Reset</u> If MCU-LINK ribbon cable or VCOM wires are in circuit, the MCU-Link must be powered-on via its USB to PC interface. <i>Users should keep the debugger USB cable connected and powered from development PC.</i>
2	Host USB port maps to kUSB_ControllerEhci1, but all NXP SDK host USB examples map to kUSB_ControllerEhci0	<u>USB Controller #</u> <i>Users need to edit this setting in api.h of imported SDK USB host application examples.</i>
3	HyperFlash lock-up during development (due to error by the developer) is not prevented	<u>Hyperflash Recovery</u> <i>See Hyperflash notes in Sections 6.6.3 - 6.6.4</i>
4	Modem UART RTS and CTS signals are not mapped correctly to the 40-pin connector	<u>Modem Flow Control</u> <i>Use XON/XOFF software flow-control on this version of hardware</i>

14 Cautionary Notes

ESD - Handling precautions for ESD-vulnerable electronic equipment are strongly recommended. It is advised to touch the metal housing of Ethernet or USB connectors, prior to touching any other part of the PCB.

Connectors - Use care when inserting/unplugging cables, especially with USB-C and audio jack connectors. Finger-support is advised to brace surface mount connectors against excessive lateral force.

MIPI Connectors - The locking mechanism of MIPI-DSI and MIPI-CSI ribbon cable connectors are relatively fragile and should be handled with care. Make sure of alignment and use minimal force when locking.

OTP eFuses – For security reasons, the on-chip OTP fuses in the RT1176 are one-time programmable. During application development, there should never be a need to program these fuses. Two boot modes are supported via selection jumper (J19). Programming of OTP eFuses will only be necessary when deploying this board in an end-product. It is the user's responsibility to be absolutely certain of their requirements before OTP programming and to not program the fuses by accident.

Note that Avnet accepts no liability and will not replace boards that have been:

- Damaged by ESD or mishandling.
- Compromised through OTP eFuse programming

15 Technical Support

15.1 NXP-hosted Technical Support Resources

NXP Technical support of software enablement (eg. IDE and SDK issues) plus the broader RT family is available via NXP-hosted public community forums, eg.

<https://community.nxp.com/t5/Software-Forums/ct-p/Software>

<https://community.nxp.com/t5/i-MX-RT/bd-p/imxrt>

For assistance with in-depth debugging using the MCUXpresso IDE, parts 1-6 of the following NXP video series is of high value:

<https://www.nxp.com/design/training/advanced-debugging-with-mcuxpresso-ide-part-1-building-debugging-and-direct-flashing:TIP-ADVANCED-DEBUG-MCUXPRESSO-IDE-1>

15.2 Avnet-hosted Technical Support Resources

Avnet documents & reference designs will be available for download from MaaXBoard RT product page:

<http://avnet.me/MaaXBoard RT>

Avnet instructional tutorial blogs will also be linked to from:

<http://avnet.me/MaaXBoard RT>

16 Sales Contact Info

For further info on Avnet-designed Starter Kits, contact your local Avnet representative at:

Region	Organization	Contact Webpage	Address & Phone
North America	Avnet Americas	www.avnet.com/contact	2211 South 47th Street Phoenix, AZ 85034, USA Phone: +1-800-585-1602
EMEA	Avnet Silica	avnet-silica.com/contact	Gruber Str. 60c 85586 Poing, Germany Phone: +49-8121-77702
EMEA	EBV	ebv.com/contact	Im Technologypark 2-8 85586 Poing, Germany Phone: +49-8121-774 - 0

17 Disclaimer

MaaXBoard RT is engineered for use as a development board (to facilitate product evaluation and system-level prototyping) as well as for use as a sub-assembly in custom OEM end-products.

Avnet assumes no liability for modifications that a user chooses to make to MaaXBoard RT.

18 Safety Warnings

Safety Warnings

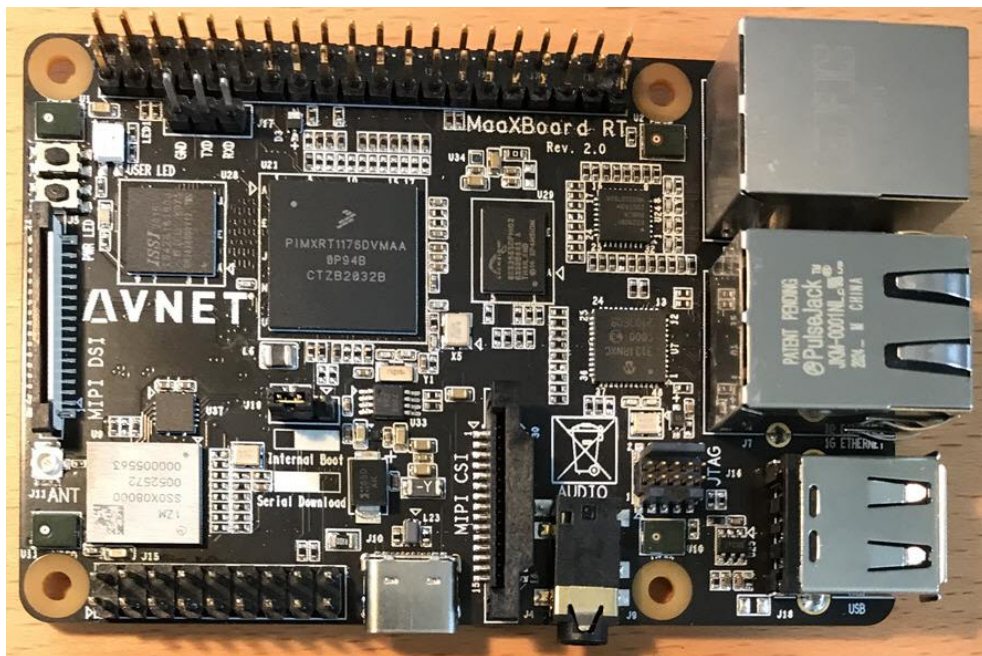
- 1) It is recommended that this product only be powered via the on board USB type-C connector, from one of the following power sources:
 - a) +5V via USB type-C cable, connected to the development computer
 - b) +5V via USB type-C cable, connected with a battery of suitable rating
 - c) +5V via USB type-C cable, connected to an external +5V, 1A DC power adaptor (a higher rating may be needed if an expansion Pi_HAT or custom board is fitted)

The external power supply shall comply with relevant regulations and standards applicable in the country of intended use.

- 2) Only compatible plug-in modules shall be connected to MaaXBoard RT.

Connection of incompatible devices may affect compliance or result in damage to the unit and void the warranty.

- 3) This product must be operated in a well-ventilated environment.
If an enclosure is used, this must provide adequate ventilation.
- 4) Do not insert or remove any expansion board or cable, without first unplugging the relevant +5V DC power source
- 5) Ambient operating temperature when using MaaXBoard RT shall not exceed the range of:
-30C to +85C



FCC Warning

This device complies with part 15 of the FCC rules. Operation is subject to the following two conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

Changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

NOTE: This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

Radiation Exposure Statement

This equipment complies with FCC radiation exposure limits set forth for an uncontrolled environment. This equipment should be installed and operated with minimum distance 20cm between the radiator and your body.

ISED Statement

English: This device complies with Industry Canada license-exempt RSS standard(s).

Operation is subject to the following two conditions: (1) This device may not cause interference, and (2) This device must accept any interference, including interference that may cause undesired operation of the device.

The digital apparatus complies with Canadian CAN ICES-3 (B)/NMB-3(B).

- French: Le présent appareil est conforme aux CNR d'Industrie Canada applicables aux appareils radio exempts de licence. L'exploitation est autorisée aux deux conditions suivantes: (1) l'appareil ne doit pas produire de brouillage, et (2) l'utilisateur de l'appareil doit accepter tout brouillage radioélectrique subi, même si le brouillage est susceptible d'en compromettre le fonctionnement.

This radio transmitter (ISED certification number: IC: 21571-MAAXBOARDRT) has been approved by Industry Canada to operate with the antenna types listed with the maximum permissible gain indicated. Antenna types not included in this list, having a gain greater than the maximum gain indicated for that type, are strictly prohibited for use with this device.

Le présent émetteur radio (ISED certification number: IC: 21571-MAAXBOARDRT) a été approuvé par Industrie Canada pour fonctionner avec les types d'antenne énumérés ci-dessous et ayant un gain admissible maximal. Les types d'antenne non inclus dans cette liste, et dont le gain est supérieur au gain maximal indiqué, sont strictement interdits pour l'exploitation de l'émetteur.

Radiation Exposure Statement

This equipment complies with Canada radiation exposure limits set forth for an uncontrolled environment. This equipment should be installed and operated with minimum distance 20cm between the radiator & your body.

Déclaration d'exposition aux radiations

Cet équipement est conforme Canada limites d'exposition aux radiations dans un environnement non contrôlé. Cet équipement doit être installé et utilisé à distance minimum de 20cm entre le radiateur et votre corps.

Caution:

- (i) The device for operation in the band 5150–5250 MHz is only for indoor use to reduce the potential for harmful interference to co-channel mobile satellite systems;
 - (ii) For devices with detachable antenna(s), the maximum antenna gain permitted for devices in the bands 5250-5350 MHz and 5470-5725 MHz shall be such that the equipment still complies with the EIRP limit;
 - (iii) For devices with detachable antenna(s), the maximum antenna gain permitted for devices in the band 5725-5850 MHz shall be such that the equipment still complies with the EIRP limits specified for point-to-point and non-point-to-point operation as appropriate; and
- Operations in the 5.25-5.35GHz band are restricted to indoor usage only.

Avertissement:

- (i) les dispositifs fonctionnant dans la bande de 5150 à 5250 MHz sont réservés uniquement pour une utilisation à l'intérieur afin de réduire les risques de brouillage préjudiciable aux systèmes de satellites mobiles utilisant les mêmes canaux;
- (ii) pour les dispositifs munis d'antennes amovibles, le gain maximal d'antenne permis pour les dispositifs utilisant les bandes de 5250 à 5350 MHz et de 5470 à 5725 MHz doit être conforme à la limite de la p.i.r.e.;
- (iii) pour les dispositifs munis d'antennes amovibles, le gain maximal d'antenne permis (pour les dispositifs utilisant la bande de 5725 à 5850 MHz) doit être conforme à la limite de la p.i.r.e. spécifiée pour l'exploitation point à point et l'exploitation non point à point, selon le cas;

Les opérations dans la bande de 5.25-5.35GHz sont limitées à un usage intérieur seulement.