ModusToolbox™ run-time software reference guide

ModusToolbox™ tools package version 3.0.0

About this document

Scope and purpose

This document helps you understand what is included in the ModusToolbox™ run-time software.

Document conventions

Convention	Explanation
Bold	Emphasizes heading levels, column headings, menus and sub-menus
Italics	Denotes file names and paths.
`Courier New`	Denotes APIs, functions, interrupt handlers, events, data types, error handlers, file/folder names, directories, command line inputs, code snippets
File > New	Indicates that a cascading sub-menu opens when you select a menu item

Reference documents

Refer to the following documents for more information as needed:

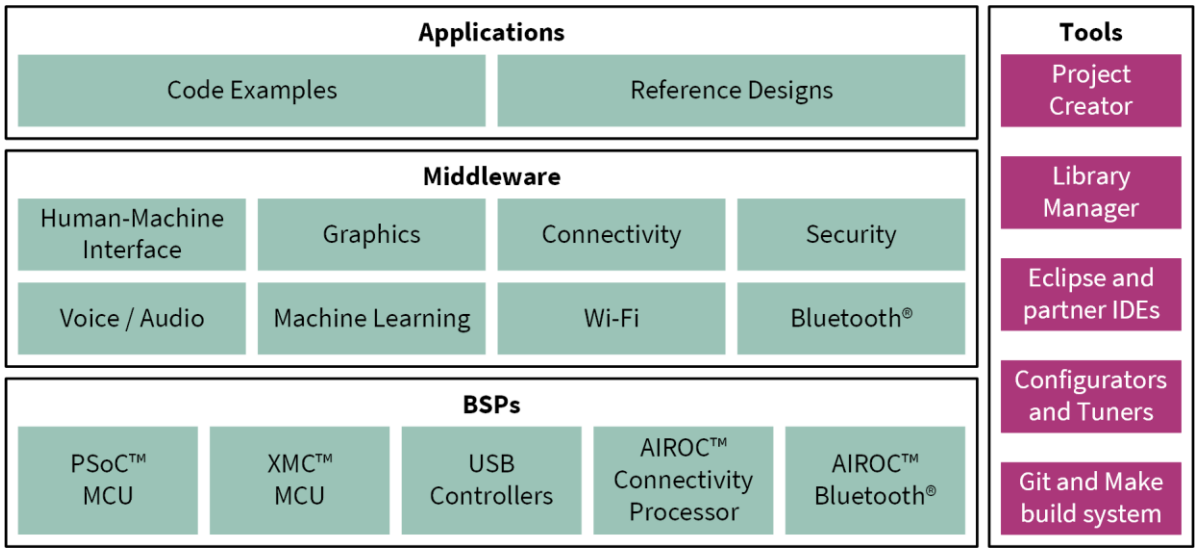
- ModusToolbox™ tools package user guide
- Project Creator user guide
- Library Manager user guide
- https://github.com/Infineon

Table of contents

Table of contents

1 Overview

The overall ModusToolbox™ ecosystem provides many types of software to help you develop applications for Infineon devices. This software includes GUIs, command-line programs, applications, middleware, and third-party software that you can use in just about any combination you need. The following shows a high-level diagram of the ecosystem.



One part of the ModusToolbox™ ecosystem is run-time software that helps you rapidly develop applications using Infineon BSPs.

This software is based on the industry standard lwIP TCP/IP stack and Mbed TLS network security. They provide core functionality including connectivity, security, firmware upgrade support, and application layer protocols like MQTT for applications that do not use commercial cloud management systems such as Arm Pelion or Amazon AWS IoT Core.

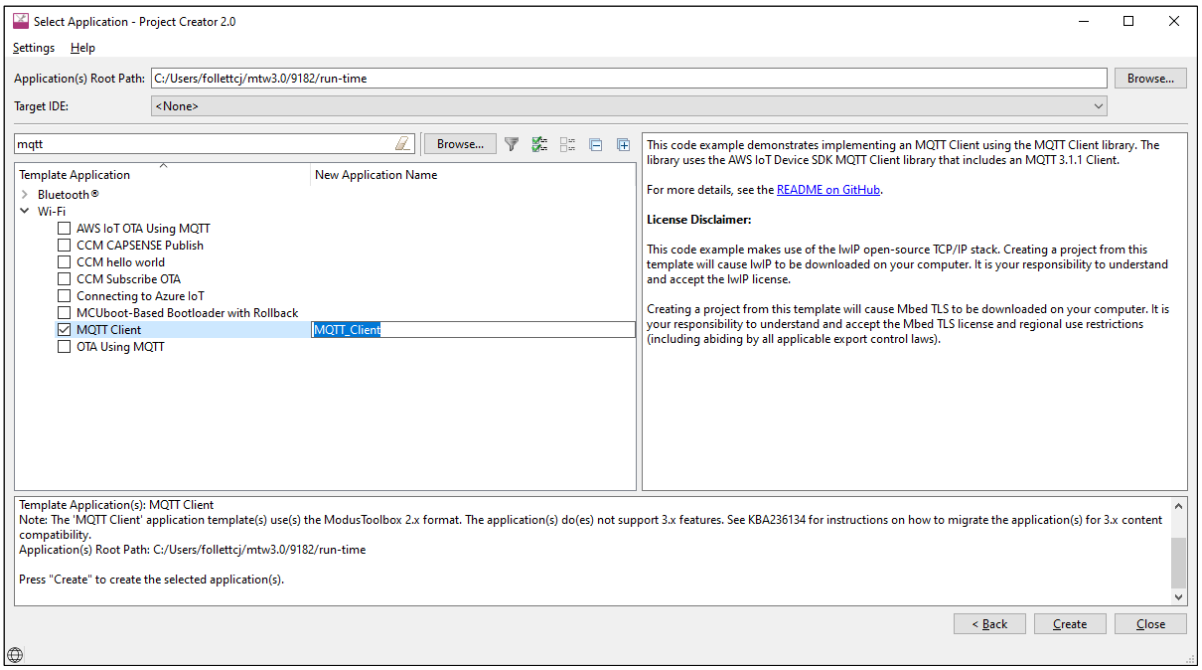
This software is intended for customers who have their own cloud device management backend, whether hosted on AWS, Google, Microsoft Azure, or another cloud infrastructure. They enable development with custom or alternative third-party cloud management approaches with a fully open, customizable, and extensible source code distribution. You can modify or extend them to match your needs.

The run-time software provides features such as the Wi-Fi Connection Manager, a Secure Socket layer, support for application layer cloud protocols, Bluetooth® Low Energy (LE) functionality, and Low Power Assist (LPA). The software currently supports TCP, MQTT, UPD, and HTTP/HTTPS client and server application layer protocols.

2 Getting started

The easiest way to get started is with an example. We provide many code examples that allow you to experiment with various features. You can get the examples by creating a ModusToolbox™ application, or by downloading them from the GitHub website:

- **Creating a ModusToolbox™ application:** Inside the ModusToolbox™ Project Creator tool, look for template applications under various categories such as Bluetooth® or Wi-Fi to find one for your needs. You can also search for a term like "mqtt" or other terms as needed and limit the number of examples that display. Refer to the Project Creator user guide for more details.

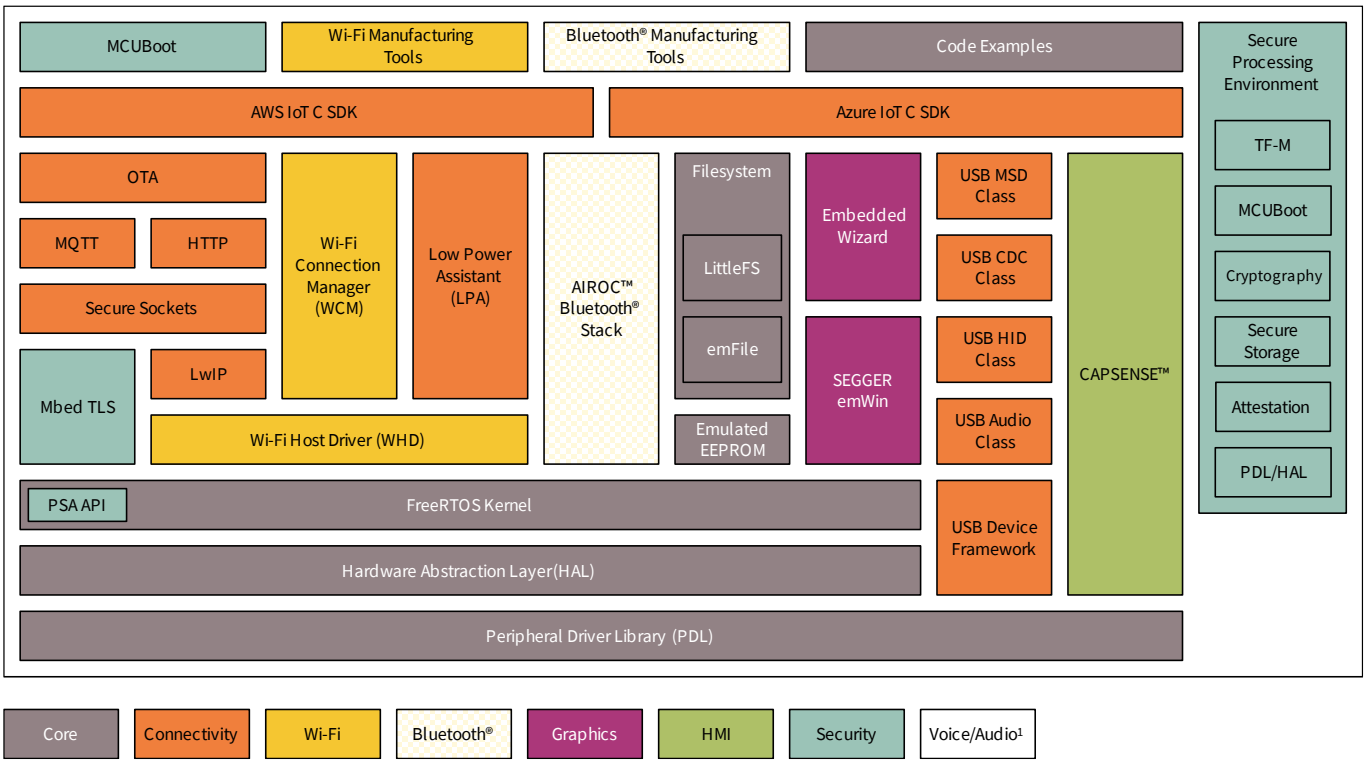


- **Downloading a code example:** Download examples directly from the GitHub website: https://github.com/Infineon/Code-Examples-for-ModusToolbox-Software

3 Run-time software descriptions and documentation

The ModusToolbox™ run-time software works together to help you easily get your IoT device connected to the cloud. Some of the libraries were written by us, while others use industry standard open source libraries. As you will see later in this document, these can be pulled into a ModusToolbox™ application using the Library Manager tool.

These libraries fit with the core PSoC™ 6 MCU libraries as shown in the following diagram. See Library Dependencies later in this document for how of these libraries are related.



[1] Voice/Audio is part of Machine Learning enablement; see https://www.infineon.com/cms/en/design-support/tools/sdk/modustoolbox-software/modustoolbox-machine-learning/ for more details.

All libraries are available as GitHub repositories. These "repos" contain source files, readme files, and documentation such as an API reference. When you include a library in your ModusToolbox™ application, the repository is downloaded into a shared directory next to the application directory. See the ModusToolbox™ tools package user guide for more details about an application's structure.

The following subsections provide brief descriptions for the main libraries, as well as links to the repository where you can find more information about them.

3.1 ModusToolbox™ core drivers and middleware

3.1.1 FreeRTOS

FreeRTOS is a small operating system for microcontrollers. It is supplied as standard C source files built along with the other C files in your project. This repository contains a port of FreeRTOS kernel for PSoC™ 6 and PSoC™ 4 MCUs, as well as for AIROC™ CYW43907, CYW54907, and CYW20829. FreeRTOS API Documentation is available at FreeRTOS web page.

See https://github.com/Infineon/freertos for more details.

3.1.2 Core drivers and peripherals

The following tables provide links to the libraries and documentation for various other categories, based on the organization in the Library Manager tool.

Library Name/Link	Description	Documentation Link
core-lib	Core Library	API Reference
psoc6cm0p	Arm Cortex M0 Plus Prebuilt Image	Repository readme file
core-make	Core Make Build System	Repository readme file
mtb-hal-cat1	Hardware Abstraction Layer	API Reference
mtb-pdl-cat1	Peripheral Driver Library	API Reference
recipe-make-cat1a	PSoC™ 6 Make Build Recipe	Repository readme file
capsense	CAPSENSE™	API Reference
clib-support	C Library Support Functions	API Reference
csdadc	CAPSENSE™ ADC (voltage)	API Reference
csdidac	CAPSENSE™ IDAC (current)	API Reference
dfu	Device Firmware Update	API Reference
emeeprom	Emulated EEPROM	API Reference
emwin	SEGGER emWin Graphics Library	Overview
usbdev	USB Device Library	API Reference

3.2 ModusToolbox™ for connectivity

3.2.1 ota-update

This library provides support for over-the-air (OTA) update of the application code running on a PSoC™ 6 MCU with the CYW43012 and other connectivity devices, using Wi-Fi or Bluetooth®.

See https://github.com/Infineon/ota-update for more details.

3.2.2 MQTT

This library includes the open source AWS IoT device SDK embedded C library plus some glue to ensure it works seamlessly in any environment. It is based on MQTT client v3.1.1 and supports QoS levels 0, 1 and 2. Both secure and non-secure TCP connections can be used.

See https://github.com/Infineon/mqtt for more details.

3.2.3 aws-iot-device-sdk-for-embedded-c

The AWS IoT Device SDK for Embedded C is a collection of C99 source files that allow applications to securely connect to the AWS IoT platform. It includes an MQTT 3.1.1 client, as well as libraries specific to AWS IoT, such as Thing Shadows. It is distributed in source form and may be built into firmware along with application code. This library supersedes both the AWS IoT Device SDK Embedded C and the libraries provided with FreeRTOS.

See https://github.com/aws/aws-iot-device-sdk-embedded-C for more details.

3.2.4 aws-iot-device-sdk-port

This library contains the port layer implementation for the MQTT and HTTP Client libraries to work with the AWS-IoT-Device-SDK-Embedded-C library on PSoC™ 6 MCU based platforms with network connectivity. This library APIs are not expected to be called by application directly. See the MQTT and HTTP Client library documentation for more details. Using this library in a project will cause AWS-IoT-Device-SDK-Embedded-C library to be downloaded on your computer. It is your responsibility to understand and accept the AWS-IoT-Device-SDK-Embedded-C license.

See https://github.com/Infineon/aws-iot-device-sdk-port for more details.

3.2.5 Secure sockets

This includes network abstraction APIs for underlying lwIP network stack and MbedTLS security library. The secure sockets library eases application development by exposing a socket like interface for both secure and non-secure socket communication.

See https://github.com/Infineon/secure-sockets for more details.

3.2.6 lwIP

This is a lightweight open-source TCP/IP stack.

See this third-party website for more details: http://www.nongnu.org/lwip/2_1_x/index.html.

3.2.7 TCP and UDP

These are part of lwIP. See the following code examples for more information:

- https://github.com/Infineon/mtb-example-anycloud-tcp-server
- https://github.com/Infineon/mtb-example-anycloud-tcp-client
- https://github.com/Infineon/mtb-example-anycloud-udp-server
- https://github.com/Infineon/mtb-example-anycloud-udp-client

3.2.8 Low power assistant (LPA)

The LPA is a library and associated settings in the ModusToolbox™ Device Configurator that allow you to configure a PSoC™ 6 Host and WLAN (Wi-Fi / Bluetooth® Radio) device for optimized low-power operation. With LPA you can achieve the most aggressive power budgets by placing the host device into sleep or deep sleep modes while networks are quiet or when there is traffic that can be handled by the connectivity device.

See https://github.com/Infineon/lpa for more details.

3.2.9 HTTP server & HTTP client

The http-server library provides communication functions for HTTP (Hypertext Transfer Protocol) Server. The HTTP client library can work with the family of our connectivity chips. This library uses the AWS IoT Device SDK HTTP client library and implements the glue layer that is required for that library to work with our connectivity platforms. This library supports RESTful HTTP methods such as HEAD, GET, PUT, POST, DELETE, PATCH, CONNECT, OPTIONS, and TRACE to communicate with any HTTP Server.

See https://github.com/Infineon/http-server and https://github.com/Infineon/http-client for more details.

3.2.10 Command console

This library provides a framework to add command console support to your application, including network operations such as scan and join. Support for Wi-Fi, iPerf, and Bluetooth® Low Energy (LE) commands is bundled with this library.

See https://github.com/Infineon/command-console for more details.

3.2.11 Azure C SDK port library

This library implements the port layer for the Azure SDK for Embedded C to work on PSoC™ 6 MCU connectivity-enabled platforms.

See https://github.com/Infineon/azure-c-sdk-port for more details.

3.3 ModusToolbox™ for security

3.3.1 Trusted Firmware-M

Trusted Firmware-M is free software that provides secure world software for Arm Cortex-M processors.

See https://github.com/Infineon/trusted-firmware-m for more details.

3.3.2 MbedTLS acceleration

The MbedTLS library is an open source, portable, easy to use, readable and flexible SSL library that has cryptographic capabilities implemented for PSoC™ 6 MCUs.

We provide a library that extends MbedTLS to enable hardware-accelerated encryption on PSoC™ 6 MCUs.

See https://github.com/Infineon/cy-mbedtls-acceleration for more details.

3.3.3 MCUBoot

MCUBoot is a secure bootloader for 32-bit Microcontrollers. It defines a common infrastructure for a productized bootloader and the flash layout of the system as well as providing a continuation of the Root of Trust (RoT) for image verification.

See https://github.com/mcu-tools/mcuboot for more details.

3.4 ModusToolbox™ for Wi-Fi

3.4.1 Enterprise security

This library provides the capability for our best in class Wi-Fi enabled PSoC™ 6 devices to connect to enterprise Wi-Fi networks. This library implements a collection of the most commonly used Extensible Authentication Protocols (EAP) that are used in enterprise networks.

See https://github.com/Infineon/enterprise-security for more details.

3.4.2 Wi-Fi Middleware Core (wifi-mw-core)

The wifi-mw-core library bundles the core libraries that any Wi-Fi application needs. These include FreeRTOS, lwIP TCP/IP stack, Mbed TLS, Wi-Fi host driver (WHD), secure sockets interface, configuration files, and associated code to bind these components together.

See https://github.com/Infineon/wifi-mw-core for more details.

3.4.3 Wi-Fi Connection Manager (WCM)

The WCM includes the wifi-mw-core library by default and provides easy to use APIs to establish and monitor Wi-Fi connections on our devices that support Wi-Fi connectivity. The WCM library also provides additional features such as Wi-Fi Protected Setup (WPS).

See https://github.com/Infineon/wifi-connection-manager for more details.

3.4.4 Wi-Fi Host Driver (WHD)

The WHD is an independent, embedded driver that provides a set of APIs to interact with our WLAN chips. This firmware product is easily portable to any embedded software environment, including popular IoT frameworks like Mbed OS, Amazon FreeRTOS, etc. So, the WHD includes hooks for RTOS and TCP/IP network abstraction layers. WHD supports the CYW43012, CYW4343W and CYW43438 Wi-Fi + Bluetooth® combo devices, as well as the CYW43907/CYW54907 Wi-Fi modules.

See https://github.com/Infineon/wifi-host-driver for more details.

3.4.5 wpa3-external-supplicant

The WPA3 External Supplicant supports WPA3 SAE authentication using HnP (Hunting and Pecking Method) using RFC, as well as H2E (Hash to Element Method) using RFC following 802.11 spec 2016.

See https://github.com/Infineon/wpa3-external-supplicant for more details.

3.5 ModusToolbox™ for Bluetooth®

3.5.1 Bluetooth™ libraries

In addition to the great Wi-Fi support, you can use the Bluetooth® LE functionality in the AIROC™ CYW43xxx combo device to enable Bluetooth® LE for your device. For example, it can easily be used to enable Wi-Fi onboarding so that you can safely and quickly connect your device to a Wi-Fi network using Bluetooth® LE to select the network and enter the password.

The Library Manager name for the Hosted Stack library is *bluetooth-freertos*. The Hosted Stack library includes the BTSTACK library automatically.

See https://github.com/Infineon/bluetooth-freertos and https://github.com/Infineon/btstack for more details.

3.5.2 btstack-integration

Support for PSoC™ 6 Bluetooth® LE Sub-System (BLESS) Controller with our AIROC™ BTSTACK. Newer designs will use this functionality, and older designs will be migrated over time as well.

See https://github/Infineon/btstack-integration for more details.

3.5.3 Smart CoEX

Bluetooth® LE and Wi-Fi operate in the same band, and in some devices, share a single radio. This can cause data transmission on one interface to interfere with the other. Such interference can impact the performance of both Wi-Fi and Bluetooth® LE. In order to avoid such interference, coexistence (CoEX) configurations and algorithms are introduced in the underlying WLAN and Bluetooth® stacks. These configurations can be tuned from the application via the configurator to test and improve the performance of underlying Wi-Fi and Bluetooth® LE traffic when they are operated simultaneously.

This library provides an API that enables the application to configure the WLAN and Bluetooth® LE stack with the parameters that were set through the configurator.

See https://github/Infineon/smartcoex for more details.

3.5.4 ble-mesh

The Bluetooth LE Mesh library provides APIs for application developers to use and create mesh applications.

See https://github.com/Infineon/ble-mesh for more details.

3.6 Connectivity middleware tools

This section list and describes various tools used for testing and bring-up.

3.6.1 Wi-Fi Bluetooth® tester

This application integrates the Command Console Library including Wi-Fi iPerf and Bluetooth® LE functionality. You can use this application to characterize the Wi-Fi/ Bluetooth® LE functionality and performance.

See https://github.com/Infineon/mtb-anycloud-wifi-bluetooth-tester for more details.

3.6.2 WLAN manufacturing test application

The WLAN Manufacturing Test Application is used to validate the WLAN firmware and radio performance of various Wi-Fi chips.

See https://github.com/Infineon/mtb-anycloud-wifi-mfg-tester for more details.

- WLAN Manufacturing Test Middleware

  The WLAN Manufacturing Test Middleware application is used to validate the WLAN firmware and radio performance of Wi-Fi devices.

  See https://github.com/Infineon/wifi-mfg-test for more details.

3.6.3 Wi-Fi cert tester tool

This tool is used for Wi-Fi Certification of 11n, PMF, WPA3 and 11AC. The Wi-Fi cert tester tool uses the command console asset to initialize and invokes wifi-cert middleware init function.

https://github.com/Infineon/mtb-anycloud-wifi-cert-tester

- Wi-Fi Cert Middleware for All SDKs

  The Wi-Fi Cert middleware provides an easy way to test WFA cert test such as 11n, PMF, WPA3 and 11AC across all SDKs providing a common interface for all SDKs.

  See https://github.com/Infineon/wifi-cert for more details.

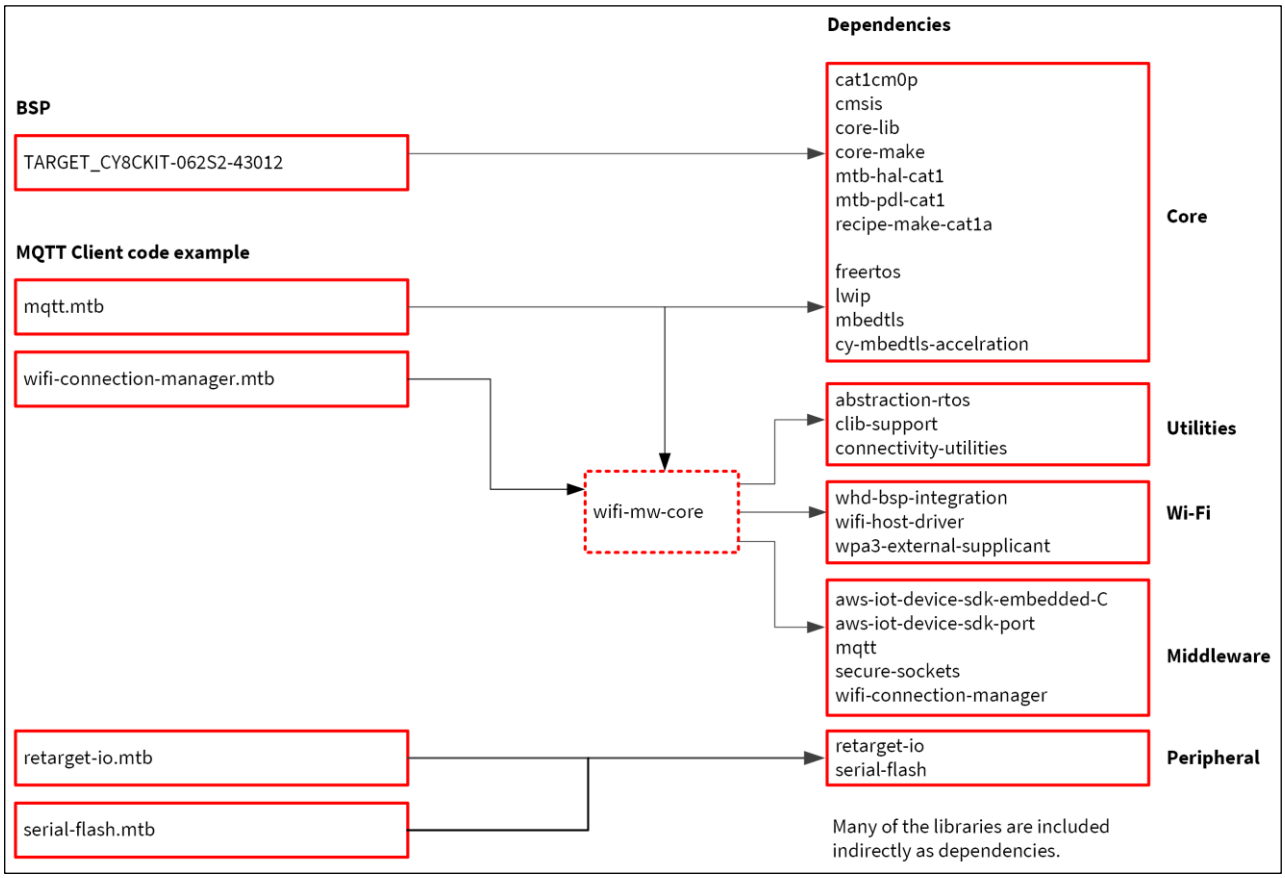
3.6.4 Bluetooth® MFG tester

The Bluetooth® Manufacturing Test Application is used to validate the Bluetooth® Firmware and RF performance of Bluetooth® BR/EDR/LE devices.

https://github.com/Infineon/mtb-anycloud-bluetooth-mfg-tester

4 Library dependencies

When you include certain libraries such as wifi-mw-core or WCM, there are dependencies on other libraries to ensure everything works correctly. As an example, the following shows some of the libraries for the MQTT Client code example and where they come from.

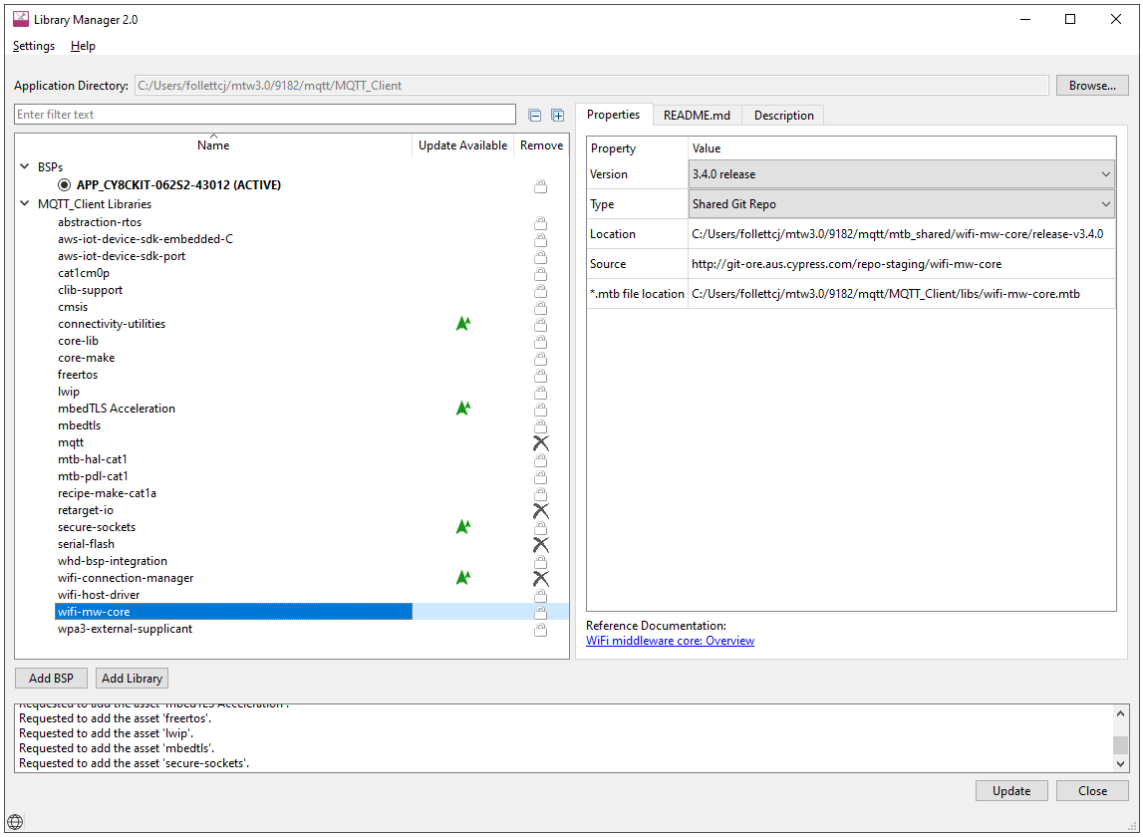


The code example includes four *.mtb* files directly. After the application has been created and processed, it contains numerous different libraries. See Adding Libraries later in this section for more details.

Using the wifi-mw-core library ensures you always have the essential Wi-Fi and networking libraries, plus good default configurations, in every cloud connected application you create.

4.1 Library manager

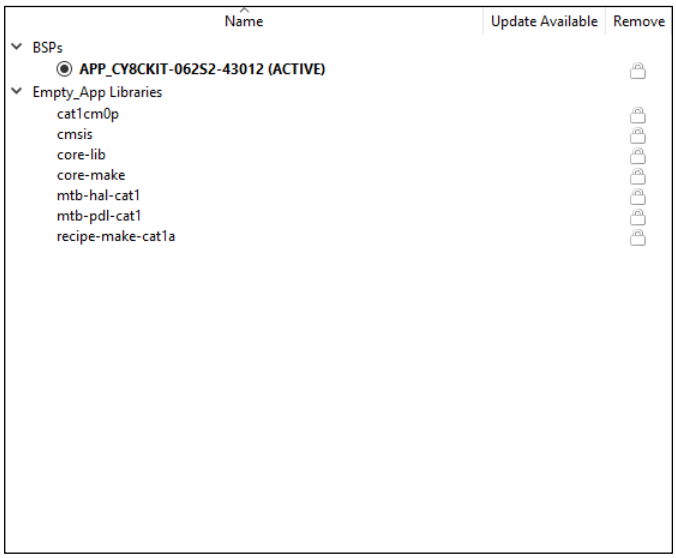
The ModusToolbox™ Library Manager tool allows you to add and remove board support packages (BSPs) and libraries, as well as select specific versions of BSPs and libraries. Refer to the Library Manager user guide for more details. As you can see in the following image, the MQTT Client code example application already includes several libraries.

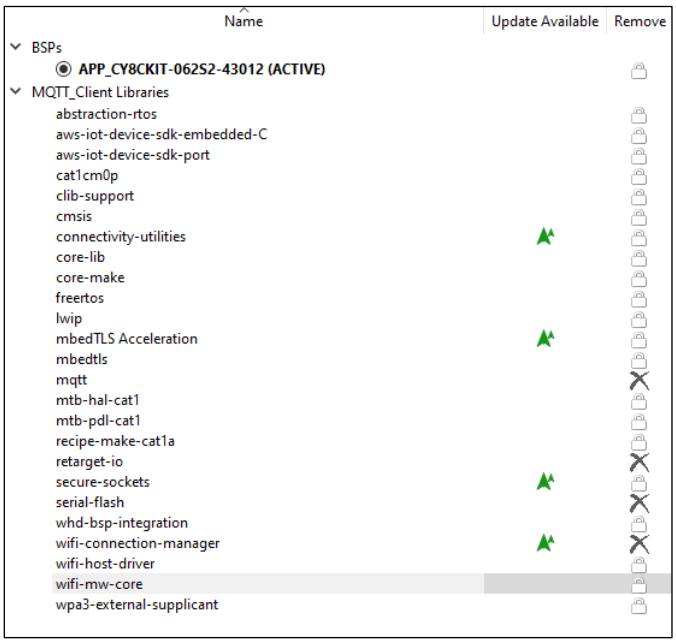


4.2 Adding libraries

As noted earlier, when the ModusToolbox™ build system encounters a *.mtb* file, it adds that library of code to the application. That library may contain additional *.mtb* files for dependent libraries. The ModusToolbox™ build system parses the *.mtb* files recursively so that all dependent libraries are added automatically. You do not need to know the dependencies.

The following images show the libraries included as part of an Empty PSoC™ 6 application before and after adding libraries.





When adding only a few libraries to the application, several other libraries are automatically included as well. The added libraries have their own *.mtb* files for the wifi-mw-core library, which in turn has *.mtb* files for the libraries it depends upon. The same paradigm applies to various libraries that have dependencies. If you add a library that requires other libraries, they will be added automatically.

Dependencies are not necessarily bi-directional. The LPA, wifi-mw-core, and mbedTLS libraries provide good examples to illustrate this concept. The wifi-mw-core does not depend upon the LPA. The LPA is optional; so, adding wifi-mw-core to an application does not add the LPA library. However, the LPA requires the wifi-mw-core. Therefore, when you add the LPA to an application, the wifi-mw-core (and all its dependencies) are added automatically. Similarly, the wifi-mw-core depends upon the MbedTLS library. However, the MbedTLS library does not depend upon the wifi-mw-core. It has no dependencies of its own.

5 Library configuration files

Some libraries provide configuration header file templates, such as the *FreeRTOSConfig.h* file. When adding a library to an application, copy the configuration file to the top-level application directory where you can edit it to customize the library. Even though there may be multiple files with the same name in an application, the ModusToolbox™ build system automatically picks the one at the top-level.

If you want to put the application-specific configuration files in a different location, you must specify the path to that directory (relative to the application's root directory) in the application's *Makefile* `INCLUDE` variable. The build system includes files in that path before it searches through the application's hierarchy.

The following are some examples of configuration files that you may need:

5.1 Wi-Fi middleware core

The template files are in the *wifi-mw-core/configs* subdirectory. See also the Quick Start in the library's README.md file (https://github.com/Infineon/wifi-mw-core).

- *FreeRTOSConfig.h*: Settings for FreeRTOS.

  Use this file as a template for FreeRTOS configuration instead of the one from the FreeRTOS library. It has some modifications specific to use with other ModusToolbox™ run-time software.

- *mbedtls_user_config.h*: Settings for Mbed TLS.

  In addition to copying this file, you must also configure the macro `MBEDTLS_USER_CONFIG_FILE` to specify the file's location and add the macro to the list of DEFINES in the application's *Makefile*. For example, if you put the file in the top level, you would include this in the *Makefile*:

  ```
  DEFINES+=MBEDTLS_USER_CONFIG_FILE='"mbedtls_user_config.h"'
  ```

  Note that many code examples use this format instead:

  ```
  MBEDTLSFLAGS = MBEDTLS_USER_CONFIG_FILE='"mbedtls_user_config.h"'

  DEFINES+=$(MBEDTLSFLAGS)
  ```

- *lwipopts.h*: Settings for lwIP. Applications may choose to modify this file in order to optimize memory consumption based on the Wi-Fi characteristics of the application.

5.1.1 Optimizing smaller memory devices

Depending on your application or device size, you may need to reduce flash and RAM usage. The configuration files included with the lwIP and MbedTLS libraries provide various parameters to enable or disable features and optimize your application's size. For example, the `MBEDTLS_SSL_SRV_C` parameter enables code when the device is expected to function as a SSL/TLS server. If you don't need this feature, you can disable it to save flash:

```
#undef MBEDTLS_SSL_SRV_C
```

Refer to the Wi-Fi Middleware Core documentation for specific details about the parameters to adjust.

5.2 MQTT

The template file is in the *mqtt/include* subdirectory. See also the Quick Start in the library's README.md file (https://github.com/Infineon/mqtt).

- *core_mqtt_config.h*: Settings for MQTT.

5.3 OTA

The template file is in the ota-update/configs subdirectory. See also the library's *README.md* file (https://github.com/Infineon/ota-update).

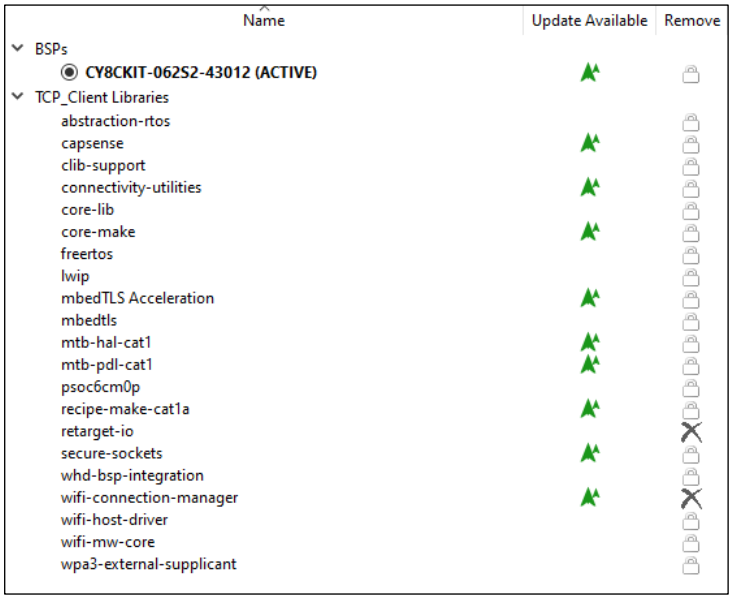
- *cy_ota_config.h*: Settings for OTA.

6 Working with examples

As described previously, the best way to learn is to download some examples and work through them. The examples include *README.md* files that guide you through the process to create and configure the application.

If you're already familiar with ModusToolbox™ software, then create the TCP Client example using your normal process. If you're new to ModusToolbox™ software, use the Eclipse IDE for ModusToolbox™. Refer to the Eclipse IDE quick start guide as needed.

After creating the application, open the Library Manager and notice that several of the libraries discussed in this guide are already present.

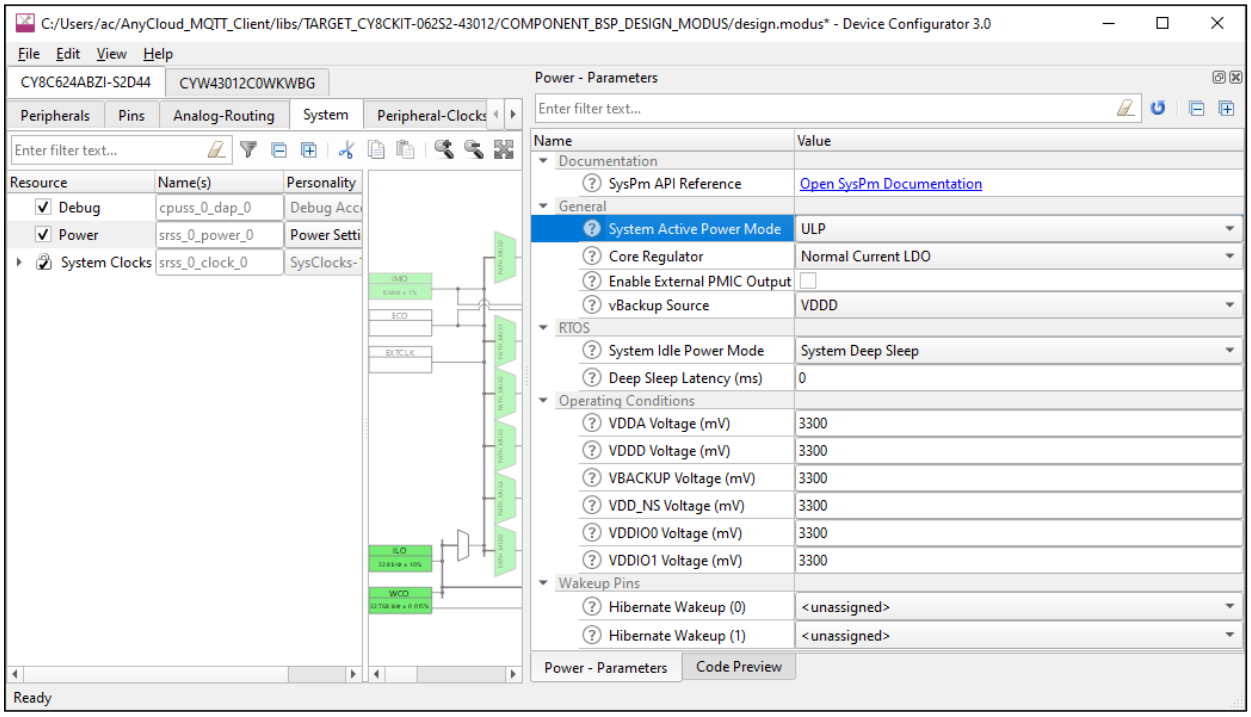


Then, create another application from an example, such as WLAN Low Power. Notice for this application that the LPA library is included.

Experiment with a few examples, and notice the differences in the libraries included, as well as various configuration options set for them. Obviously, review the code as well.

7 Adding and configuring low power

The WLAN Low Power example is one of several that demonstrate the low power features in ModusToolbox™ software. Among the low power features are the power settings accessible in the Device Configurator.



There are settings for the PSoC™ 6 MCU and the connectivity combo device. For more details about how to configure low power, refer to the LPA Guide here:

https://Infineon.github.io/lpa/lpa_api_reference_manual/html/index.html

8 Adding and configuring Bluetooth®

As you have probably already discovered, there are several Bluetooth® examples as well. For example, the Wi-Fi Onboarding Using Bluetooth® LE shows how to use Bluetooth® on the combo device to help connect the Wi-Fi device to an access point. It also shows how to enable low-power modes on both the Wi-Fi and Bluetooth® devices. For more details, refer to the example's *README.md* file.

Revision history

Version	Date	Description
**	2020-06-29	New document.
*A	2020-10-05	Updated the libraries. Added optimizing smaller memory devices. Added supported devices.
*B	2020-10-23	Updated diagram to remove ble-ota.
*C	2020-12-20	Added http and CoEX.
*D	2021-05-11	Updated for version 1.4 changes.
*E	2021-05-24	Updated for version 1.4.1 changes.
*F	2021-10-05	Updated for version 1.5 changes.
*G	2021-12-15	Updated for January 2022.
*H	2022-04-05	Updated for April 2022. Changed the term any cloud to run-time software.
*I	2022-08-05	Updated for June 2022.
*J	2022-10-19	Updated for October 2022.

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

Edition 2022-10-19
Published by
Infineon Technologies AG
81726 Munich, Germany

Do you have a question about this document?
Email: erratum@infineon.com

Document reference
002-30738 Rev. *J

IMPORTANT NOTICE
The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics (“Beschaffenheitsgarantie”) .

With respect to any examples, hints or any typical values stated herein and/or any information regarding the application of the product, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation warranties of non-infringement of intellectual property rights of any third party.

In addition, any information given in this document is subject to customer’s compliance with its obligations stated in this document and any applicable legal requirements, norms and standards concerning customer’s products and any use of the product of Infineon Technologies in customer’s applications.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer’s technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on the product, technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies office (**www.infineon.com**).

WARNINGS
Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies’ products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.