ModusToolbox™ run-time software reference guide

ModusToolbox™ tools package version 3.2.0

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

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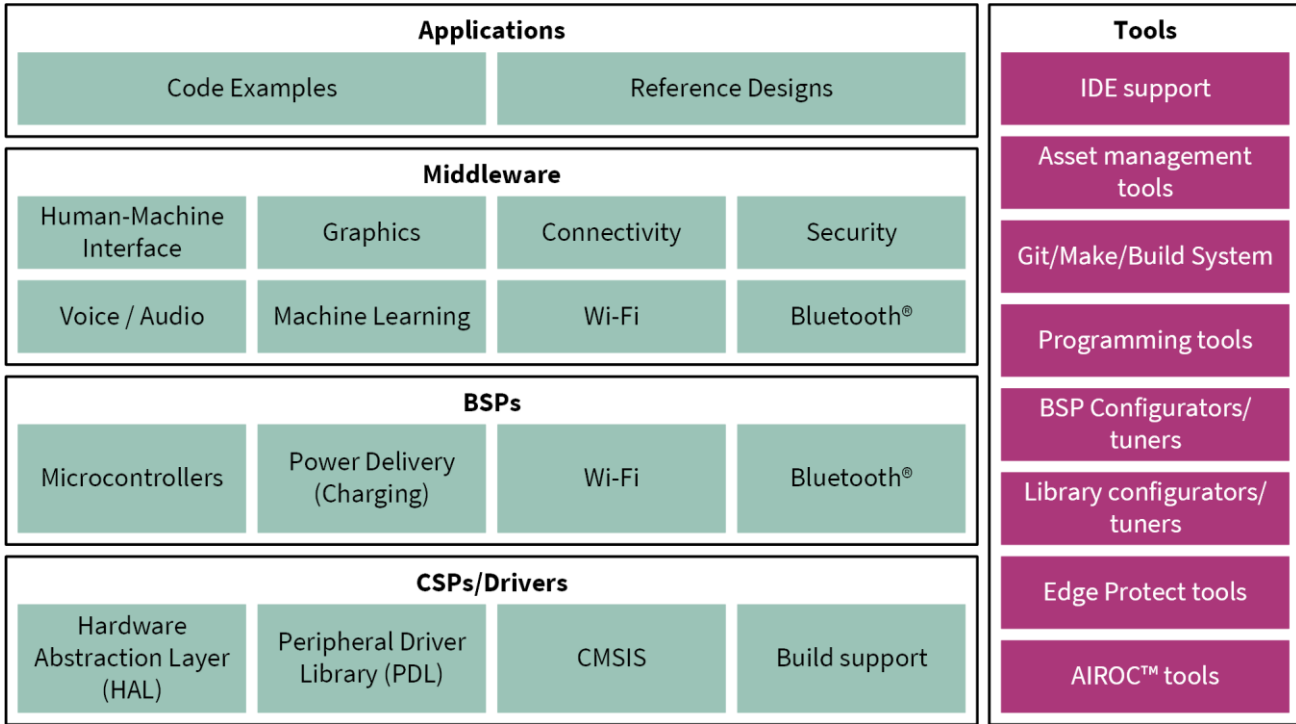
One part of the ModusToolbox™ ecosystem is run-time software that helps you rapidly develop applications using Infineon BSPs.

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, UDP, 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.
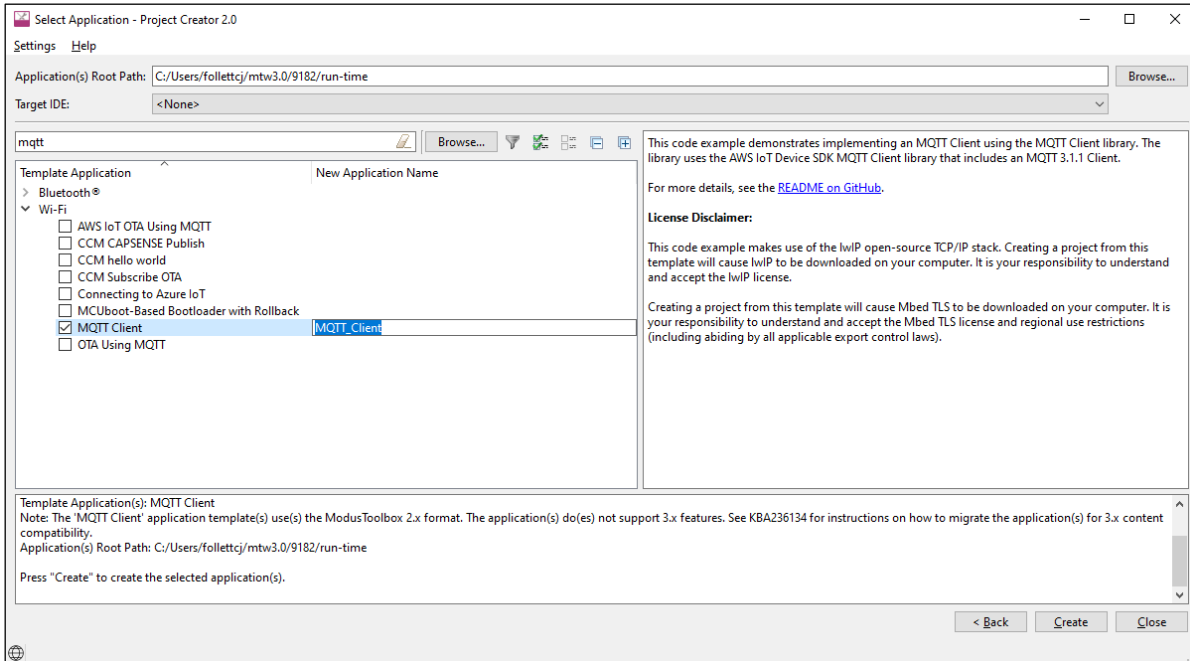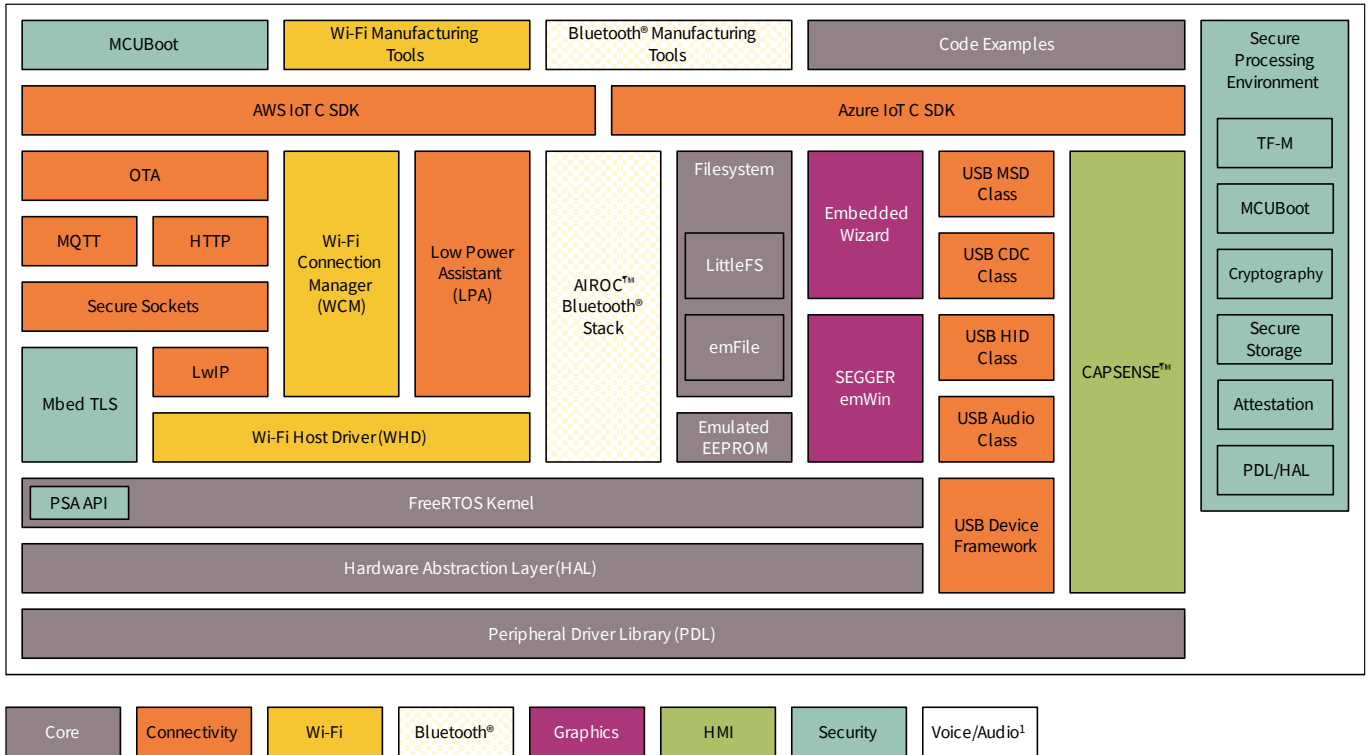

- **Downloading a code example:** Download examples directly from the GitHub website:

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 various libraries as shown in the following diagram. See Library Dependencies later in this document for how of these libraries are related.

MCUBoot	Wi-Fi Manufacturing Tools	Bluetooth® Manufacturing Tools	Code Examples	Secure Processing Environment

Core, Connectivity, Wi-Fi, Bluetooth®, Graphics, HMI, Security, Voice/Audio[1]

[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

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

3.1.1 General

Library	Details	Docs
abstraction-rtos	Abstraction-RTOS has been created by Infineon to allow the flexibility of Infineon created software to run within any RTOS. It may be recommended by Developers providing libraries to use the abstraction RTOS; however, it is not recommended to use this as the main RTOS interface. It is recommended that developers should use the APIs of the chosen RTOS (FreeRTOS/Azure RTOS/Zephyr RTOS).	API reference
clib-support	The CLib FreeRTOS support library provides the necessary hooks to make C library functions such as malloc and free thread safe.	API reference
cmsis	Used by application or middleware to link CMSIS Core headers	README
command console	Provides a framework to add command console support to the application (or) product use cases.	API reference
core-lib	Header files that declare basic types and utilities (such as result types or ASSERT) that can be used by multiple BSPs.	API reference
csdadc	Enables the ADC functionality of the CAPSENSE™ Sigma-Delta (CSD) hardware block. Useful for devices that do not include other ADC/IDAC options.	API reference
csdidac	The same, for IDAC functionality.	API reference
emeeprom	The Emulated EEPROM library provides an API to manage an emulated EEPROM in flash. It has support for wear leveling and restoring corrupted data from a redundant copy.	API reference
emfile	A FAT16/32 filesystem for embedded systems supporting SPI NOR flash and SD card.	User guide
emusb-device	Enables easy integration of USB functionality into an embedded system.	API reference
emusb-host	CPU-independent USB Host stack.	API reference
freertos	FreeRTOS kernel, distributed as standard C source files with configuration header file, for use with the PSoC™ 6 MCU.	FreeRTOS webpage
freertos-posix	The Portable Operating System Interface (POSIX) is a family of standards specified by the IEEE Computer Society for maintaining compatibility between operating systems. freertos-posix implements a small subset of the POSIX threading API.	POSIX API Reference
littlefs	A little fail-safe filesystem designed for microcontrollers.	README
mtb-littlefs	Provides a set of block device drivers for use with the littlefs file system.	API reference
mtb-stl	The ModusToolbox™ Safety Test Library provides functional safety APIs to implement overall safety of a system that depends on automatic protection suitable for using in industrial environments and home appliances.	README
retarget-io	Provides a board-independent API to retarget text input/output to a serial UART on a kit	API reference
serial-flash	Provides a board-independent API to use the serial flash on a kit	API reference
usbdev	The USB Device library provides a full-speed USB 2.0 Chapter 9 specification compliant device framework.	API reference

3.1.2   CAT1: PSoC™ 6, CYW20829, TRAVEO™ II, XMC7000

Library	Details	Docs
cat1cm0p	Arm® Cortex®-M0+ prebuilt images enabling flash write services and Bluetooth® Low Energy event handling.	README
mtb-hal-cat1	The PSoC™ 6 Hardware Abstraction Layer package provides a set of APIs to initialize, configure, and use the PSoC™ 6 MCU resources using our defined Hardware Abstraction Layer.	API reference
mtb-pdl-cat1	The Peripheral Driver Library (PDL) integrates device header files, startup code, and low-level peripheral drivers into a single package.	API reference

3.1.3   CAT2: PSoC™ 4, PMG

Library	Details	Docs
mtb-hal-cat2	PSoC™ 4 Hardware Abstraction Layer package, provides a set of APIs to initialize, configure, and use the PSoC™ 4 MCU resources using the defined Hardware Abstraction Layer.	API reference
mtb-pdl-cat2	The Peripheral Driver Library (PDL) integrates device header files, startup code, and low-level peripheral drivers into a single package.	API reference

3.1.4   CAT3: XMC1000, XMC 4000

Library	Details	Docs
mtb-xmclib-cat3	The XMC™ peripheral library consists of low-level drivers and CMSIS start-up code for the XMC™ product family peripherals.	API reference

3.1.5   CAT4: CYW43907, CYW54907

Library	Details	Docs
mtb-hal-cat4	Hardware Abstraction Layer package, provides a set of APIs to initialize, configure, and use the CYW43907/CYW54907 resources using the defined Hardware Abstraction Layer.	API reference

3.1.6   CAT5: CYW5551x

Library	Details	Docs
mtb-hal-cat5	Hardware Abstraction Layer package, provides a set of APIs to initialize, configure, and use the CYW5551x resources using the defined Hardware Abstraction Layer.	API reference
mtb-pdl-cat5	The Cypress PDL simplifies software development for CAT5 family of devices. The PDL integrates device header files, startup code, and peripheral drivers into a single package.	API reference

3.2   ModusToolbox™ for connectivity

Library	Details	Docs
aws-iot-device-sdk-embedded-C	Collection of C source files under the MIT open source license that can be used in embedded applications to securely connect IoT devices to AWS IoT Core.	README.md
aws-iot-device-sdk-port	Contains the port layer implementation for the MQTT and HTTP Client libraries to work with the AWS-IoT-Device-SDK-Embedded-C library on platforms with network connectivity.	API Reference

Library	Details	Docs
azure-c-sdk-port	Implements the port layer for the Azure SDK for Embedded C to work on platforms with network connectivity.	API reference
azure-sdk-for-c	Allows small embedded (IoT) devices to communicate with Azure services.	README
connectivity-utilities	General purpose middleware connectivity utilities, for instance a linked_list or a json_parser.	API reference
golioth-firmware-sdk	A software development kit for connecting embedded devices to the Golioth IoT Cloud	External website
http-client	Provides the HTTP Client implementation that can work on platforms with Wi-Fi or Ethernet connectivity.	API reference
http-server	Provides communication functions for an HTTP server.	API Reference
lpa	The Low Power Assistant (LPA) is a library and associated settings in the ModusToolbox™ Device Configurator that allow you to configure a Host and WLAN (Wi-Fi / BT Radio) device for optimized low-power operation.	API reference
lwIP	Lightweight open-source TCP/IP stack.	External website
memfault-firmware-sdk	An SDK responsible for working with Memfault device monitoring, debugging and OTA management platform.	External website
mqtt	This library includes the open source AWS IoT device SDK embedded C library plus some glue to ensure seamless MQTT cloud connectivity.	API reference
netxduo	Advanced, industrial-grade TCP/IP network stack designed specifically for deeply embedded real-time and IoT applications.	README
netxduo-network-interface-integration	This library is an integration layer that links the NetXDuo network stack with the underlying Wi-Fi host driver (WHD).	API reference
ota-update	Provides support for Over-The-Air update of the application code running on a PSoC™ 6 MCU with AIROC™ CYW4343W, CYW43012, or CYW20829 Wi-Fi & Bluetooth® combo chip, using Wi-Fi or Bluetooth®, as well as XMC7200 devices.	API reference
ota-bootloader-abstraction	The OTA Bootloader Abstraction library has implementation for bootloader specific storage interface APIs for handling OTA upgrade images.	README
secure-sockets	The Secure Sockets library eases application development by exposing a socket like interface for both secured and non-secured socket communication.	API reference
virtual-connectivity-manager	Virtual-Connectivity-Manager (VCM) is a library that enables connectivity libraries to add multi-core support through virtualization. Virtualization allows the connectivity stack running on one core to be accessed from another core using Inter Process Communication (IPC).	API reference

3.3 ModusToolbox™ for Wi-Fi and Ethernet

Library	Details	Docs
lwip-network-interface-integration	This library is an integration layer that links the lwIP network stack with the underlying Wi-Fi host driver (WHD) and Ethernet driver. This library interacts with FreeRTOS, lwIP TCP/IP stack, Wi-Fi host driver (WHD), and Ethernet driver. It contains the associated code to bind these components together.	API Reference
ethernet-connection-manager	Ethernet Connection Manager (ECM) is a library which helps application developers to manage ethernet connectivity for XMC7200 devices.	API Reference

Library	Details	Docs
ethernet-core-freertos-lwip-mbedtls	This repo comprises core components needed for ethernet connectivity support for XMC7200 devices	README
wifi-core-freertos-lwip-mbedtls	This repo comprises core components needed for Wi-Fi connectivity support. The library bundles FreeRTOS, lwIP TCP/IP stack, mbed TLS for security, Wi-Fi host driver (WHD), wifi connection manager (WCM), secure sockets, connectivity utilities and configuration files.	README
wifi-mw-core	The Wi-Fi Middleware Core library bundles the core libraries that any Wi-Fi application needs. This has been deprecated	API reference
wifi-host-driver	The Wi-Fi Host Driver (WHD) is an independent, embedded driver that provides a set of APIs to interact with our WLAN chips.	API reference
wifi-connection-manager	The Wi-Fi Connection Manager (WCM) provides easy to use APIs to establish and monitor Wi-Fi connections on our devices that support Wi-Fi connectivity.	API reference
wifi-resources	This repo contains clm and nvram files for the supported Wi-Fi boards from associated chip set.	README
smartcoex	Provides an API to configure the coex parameters for WLAN and Bluetooth® Low Energy on PSoC™ 6 MCU based platforms with Wi-Fi & Bluetooth® combo chip.	API reference
enterprise-security	This library implements a collection of the most commonly used Extensible Authentication Protocols (EAP) used in enterprise Wi-Fi networks	API reference
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.	README
whd-bsp-integration	This library provides some convenience functions for connecting the Wi-Fi Host Driver (WHD) library to a BSP that includes a WLAN chip.	API reference
ethernet-phy-driver	Provides the Ethernet PHY-related interface APIs as required by Ethernet connection manager library to enable the completion of Ethernet-based applications on supported platforms.	API reference

3.4   ModusToolbox™ for Bluetooth®

Library	Details	Docs
ble-mesh	Provides Application Programming Interfaces (APIs) for application developers to use and create Mesh Applications.	API reference
btstack	BTSTACK is our Bluetooth® Host Protocol Stack implementation. The stack is optimized to work on our Bluetooth® controllers.	API reference Dual Mode API reference
btstack-integration	Platform adaptation layer (porting layer) between AIROC™ BT Stack and Abstraction Layers (CYHAL and CYOSAL) for different hardware platforms.	README
bless	This library is specific to devices like **CY8C6347BZI-BLD53**, with onboard Bluetooth® and no separate connectivity device. **Not recommended for new designs**.	API reference

For information about BTSDK, see https://Infineon.github.io/btsdk-docs/BT-SDK/index.html.

3.5 ModusToolbox™ for Graphics

Library	Details	Docs
emwin	SEGGER embedded graphic library and graphical user interface (GUI) framework designed to provide processor- and display controller-independent GUI for any application that needs a graphical display.	Overview

3.6 ModusToolbox™ for Security

Library	Details	Docs
trusted-firmware-m	Provides secure world software for Arm Cortex-M processors.	README
dfu	The Device Firmware Update (DFU) library provides an API for updating firmware images.	API reference
Arm Mbed TLS	A library to include cryptographic and SSL/TLS capabilities in an embedded application.	API reference
cy-mbedtls-acceleration	We provide a library that extends MbedTLS to enable hardware-accelerated encryption on PSoC™ 6 MCUs.	README
MCUBoot	Secure bootloader for 32-bits microcontrollers.	README

3.7 ModusToolbox™ for Machine Learning

Library	Details	Docs
ml-inference	A set of pre-compiled libraries which provide easy to use API's to run ML workloads on embedded platforms.	API reference
ml-middleware	Helper functions to simplify integration of ML models.	API reference
ml-tflite-micro	A pre-configured TensorFlow tflite-micro runtime library for the Infineon PSoC™ 6 platform.	README

3.8 ModusToolbox™ for Voice

Library	Details	Docs
cyberon-dspotter-lib-psoc6-cm0p	Highly-accurate and lightweight wake word engine. It enables building always-listening voice-enabled applications for the CM0p Infineon devices	External website
cyberon-dspotter-lib-psoc6-cm4	Highly-accurate and lightweight wake word engine. It enables building always-listening voice-enabled applications for the CM4 Infineon devices	

3.9 ModusToolbox™ for HMI

Library	Details	Docs
CAPSENSE™	Capacitive sensing can be used in a variety of applications and products where conventional mechanical buttons can be replaced with sleek human interfaces to transform the way users interact with electronic systems.	API reference

3.10 Connectivity middleware tools

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

3.10.1 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-wifi-bluetooth-tester for more details.

3.10.2 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-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.

3.10.3 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-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.

3.10.4 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, there are dependencies on other libraries to ensure everything works correctly. As an example, the MQTT Client 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-core-freertos-lwip-mbedtls 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.
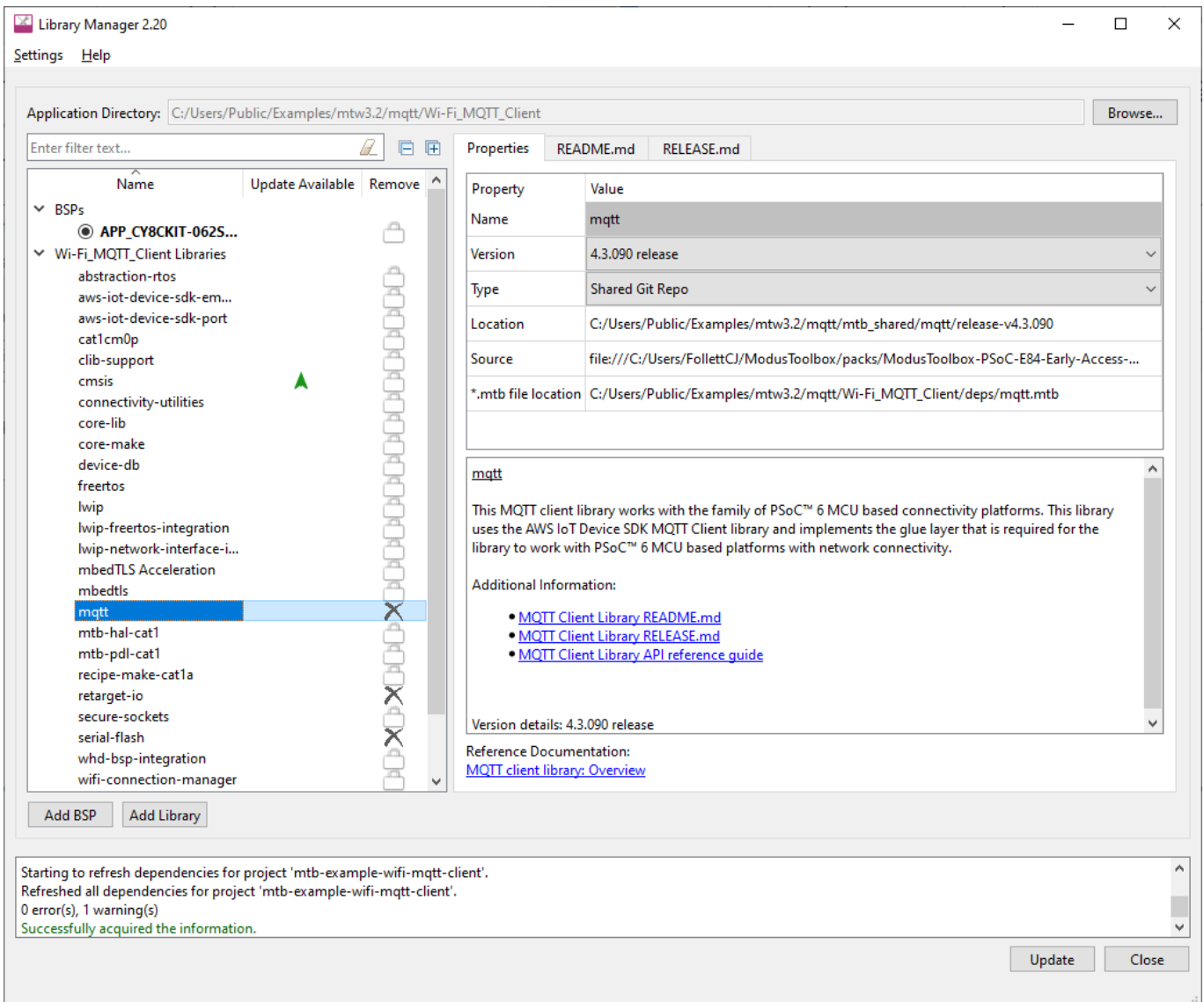
4.2 Adding libraries

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.
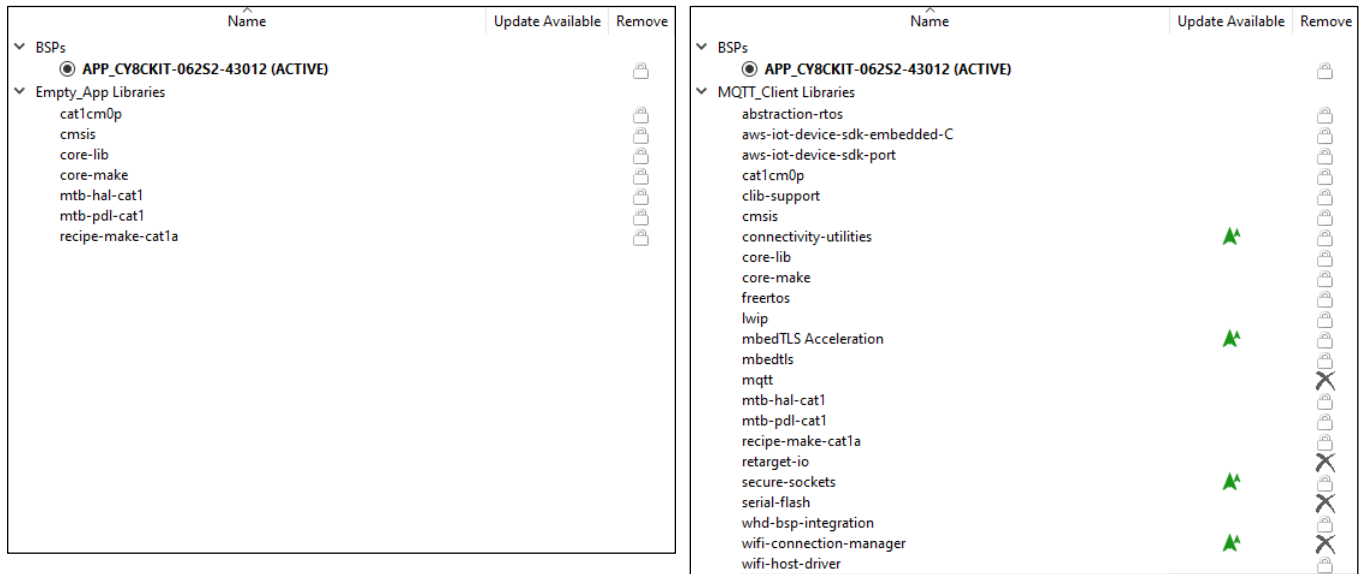
When adding only a few libraries to the application, several other libraries are automatically included as well. The added libraries have their own dependency libraries, which in turn may have dependency libraries they depend upon. The same paradigm applies to various libraries that have dependencies. If you add a library that requires other libraries, they will be added automatically.

Dependencies are not necessarily bi-directional. While some libraries have dependencies, other libraries don't -- even if they are a dependency of another library. .

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 core configuration

The Wi-Fi middleware core library has been deprecated. Use the porting guide (https://github.com/Infineon/lwip-network-interface-integration/blob/master/porting_guide.md) to migrate the application to use the lwIP network interface integration library.

The following are template files to use for configuration:

- *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

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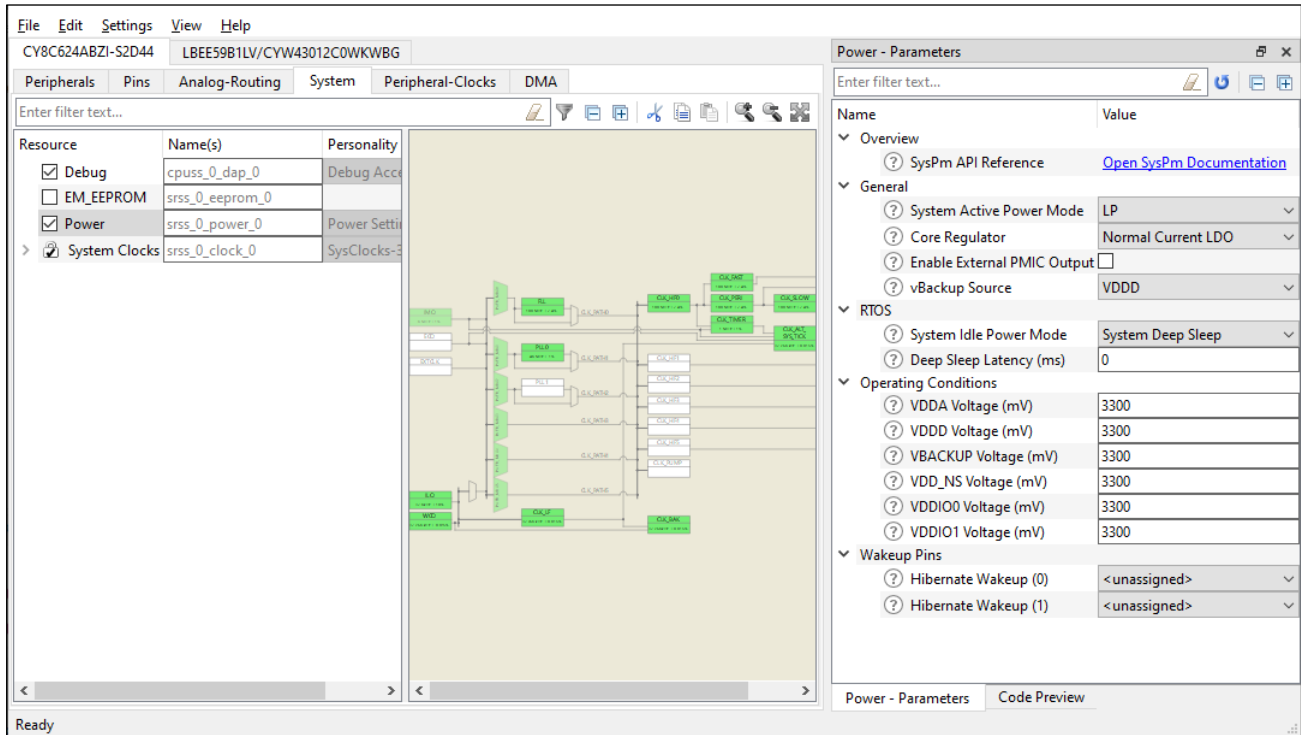
There are settings for the PSoC™ 6 MCU and the connectivity combo device. For more details about how to configure low power, refer to the LPA Guide here:

https://infineon.github.io/lpa/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.

Error! Use the Home tab to apply to the text that you want to appear here.

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.
*D	2021-05-24	Updated for version 1.4.1 changes.
*E	2021-10-05	Updated for version 1.5 changes.
*F	2021-12-15	Updated for January 2022.
*G	2022-04-05	Updated for April 2022. Changed the term any cloud to run-time software.
*H	2022-08-05	Updated for June 2022.
*I	2022-10-19	Updated for October 2022.
*J	2022-12-20	Updated for December 2022.
*K	2023-03-31	Updated for March 2023.
*L	2023-06-30	Updated for June 2023.
*M	2023-09-25	Updated for September 2023; minor changes.
*N	2024-01-31	Updated for January 2024.
*O	2024-03-25	Updated for March 2024.
*P	2024-06-26	Updated for June 2024.
*Q	2024-09-24	Updated for September 2024.

Trademarks

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

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

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.