Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

ModusToolbox® AnyCloud User Guide

AnyCloud Version 1.3

Overview

ModusToolbox is a set of tools to help you develop applications for PSoC® 6 MCU devices. These tools include GUIs, command-line programs, software libraries, and third-party software that you can use in just about any combination you need. One part of ModusToolbox is a collection of software libraries that we call AnyCloud. These libraries help you rapidly develop Wi-Fi and Bluetooth applications using connectivity combo devices, such as CYW4343W and CYW43012, with the PSoC 6 MCU.

AnyCloud is based on the industry standard lwIP TCP/IP stack and Mbed TLS network security. It provides 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.

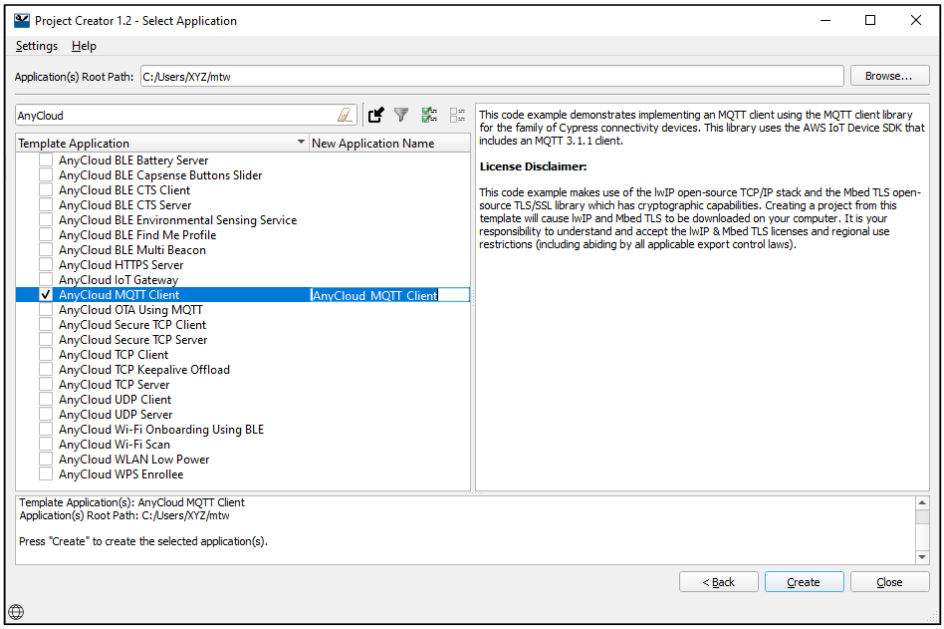
AnyCloud was built for customers who have their own cloud device management backend, whether hosted on AWS, Google, Microsoft® Azure, or another cloud infrastructure. It enables development with custom or alternative third-party cloud management approaches with a fully open, customizable, and extensible source code distribution. You can modify or extend it to match your needs.

AnyCloud provides features such as the Wi-Fi Connection Manager, a Secure Socket layer, support for application layer cloud protocols, Bluetooth Low Energy (BLE) functionality, and Low Power Assist (LPA). AnyCloud currently supports TCP and MQTT application layer protocols.

Getting Started

The easiest way to get started is with an example. Cypress provides many AnyCloud code examples. You can get them 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 starter applications whose names start with “AnyCloud.” Refer to the Project Creator Guide for more details.

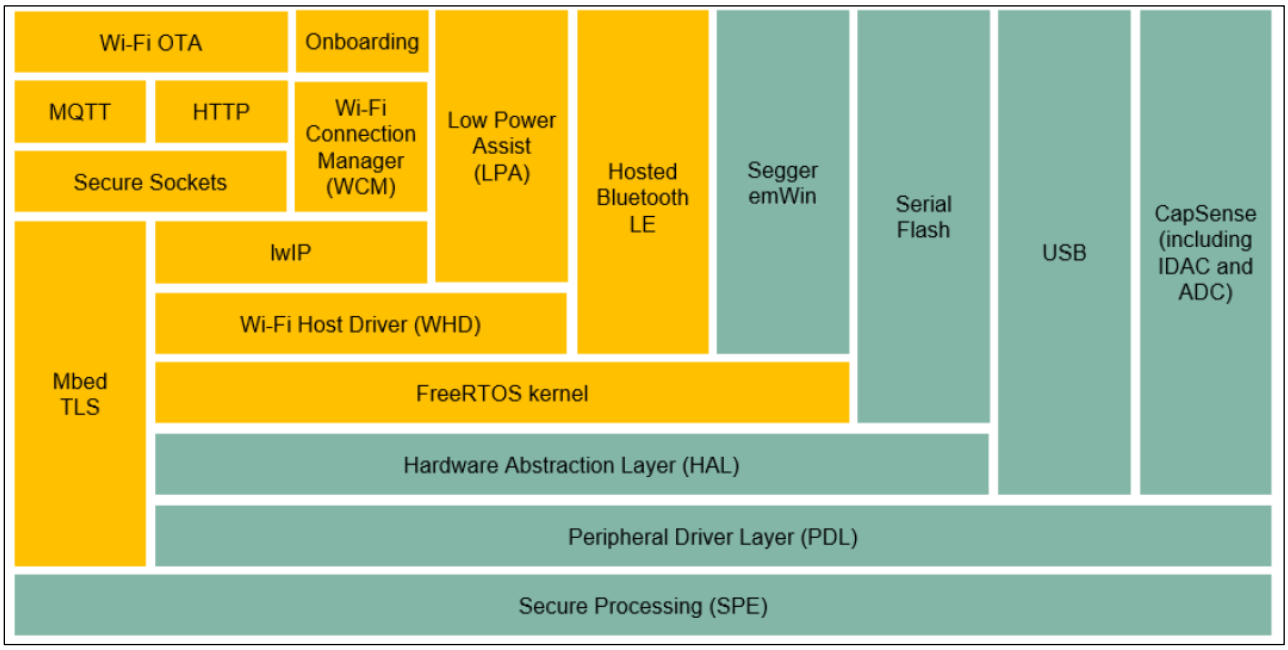


- **Downloading a Code Example:** Download examples directly from the Cypress GitHub website: https://github.com/cypresssemiconductorco?q=mtb-example-anycloud%20NOT%20Deprecated

AnyCloud Library Descriptions and Documentation

The AnyCloud solution libraries work together to help you easily get your IoT device connected to the cloud. Some of the libraries were written by Cypress, while others use industry standard open source libraries. As you will see later, these can be pulled into a ModusToolbox application using the Library Manager tool.

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



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 the application directory under the *libs* subdirectory. See the ModusToolbox User Guide for more details about an application’s structure.

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

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/cypresssemiconductorco/wifi-mw-core for more details.

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 Cypress devices that support Wi-Fi connectivity. The WCM library also provides additional features such as Wi-Fi Protected Setup (WPS).

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

Wi-Fi Host Driver (WHD)

The WHD is an independent, embedded driver that provides a set of APIs to interact with Cypress 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.

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

Over-The-Air (OTA)

This library enables applications to implement sophisticated OTA software updates from any cloud platform using custom cloud management tools.

A comprehensive code example uses OTA over MQTT with TLS to securely connect to an MQTT broker and download an update for the user's application.

See https://github.com/cypresssemiconductorco/anycloud-ota for more details.

MQTT

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

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

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/cypresssemiconductorco/secure-sockets for more details.

lwIP

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

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

MbedTLS

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

Cypress provides a library that extends MbedTLS to enable hardware-accelerated encryption on PSoC 6 devices. See https://github.com/cypresssemiconductorco/cy-mbedtls-acceleration for more details.

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 / BT 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/cypresssemiconductorco/lpa for more details.

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 Cypress 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 Cypress connectivity platforms. This library supports RESTful methods such as GET, PUT, POST, and HEAD to communicate with any HTTP Server.

Smart CoEX

Bluetooth Low Energy (BLE) 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 BLE. In order to avoid such interference, coexistence (coex) configurations and algorithms are introduced in the underlying WLAN and BT stacks. These configurations can be tuned from the application via the configurator to test and improve the performance of underlying Wi-Fi and BLE traffic when they are operated simultaneously.

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

Bluetooth Libraries

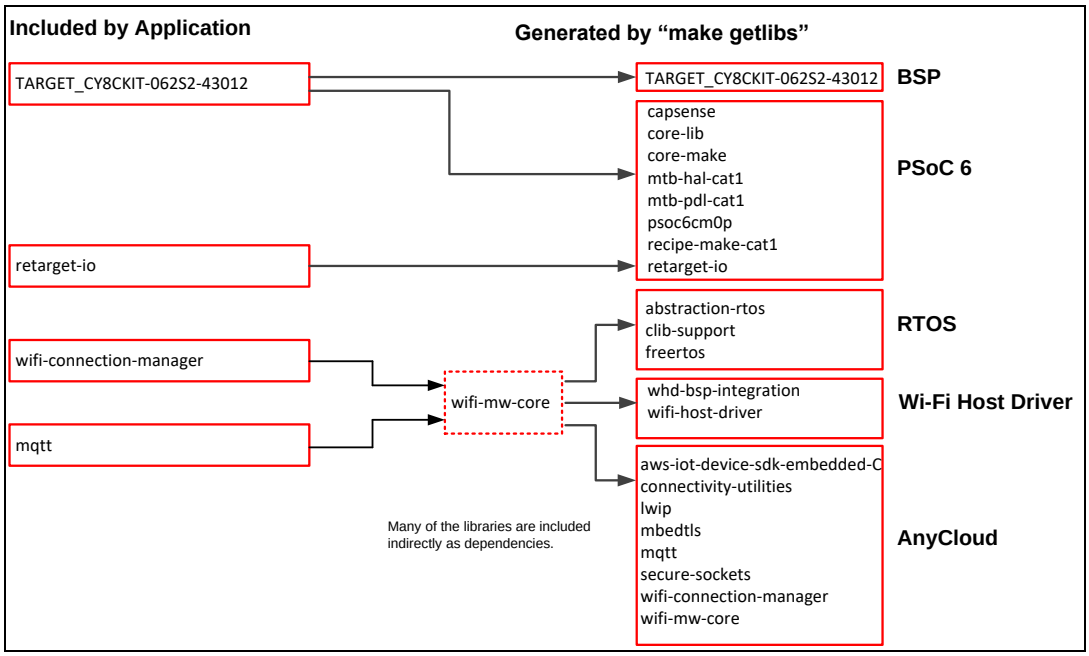
In addition to the great Wi-Fi support in AnyCloud, you can use the Bluetooth LE functionality in the 43xxx combo device to enable BLE for your device. For example, it can easily be used to enable Wi-Fi onboarding so that you can safely and quickly connect your device to a Wi-Fi network using BLE to select the network and enter the password.

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

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

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 the libraries for the AnyCloud MQTT Client code example and where they come from.

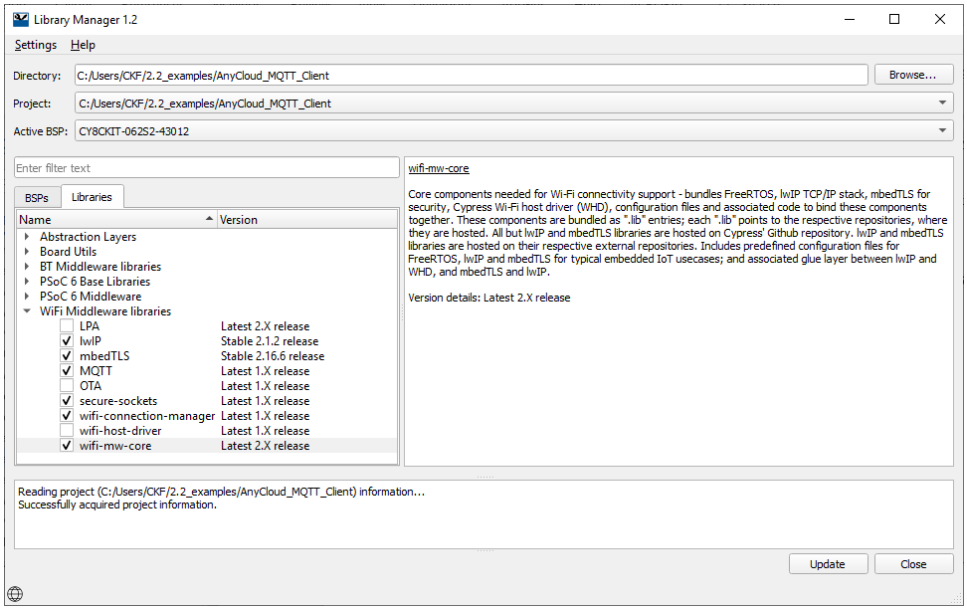


The application includes 4 .lib files directly. After the application has been created and processed, it contains 22 different libraries. See Adding Libraries later in this section for more details.

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.

Library Manager

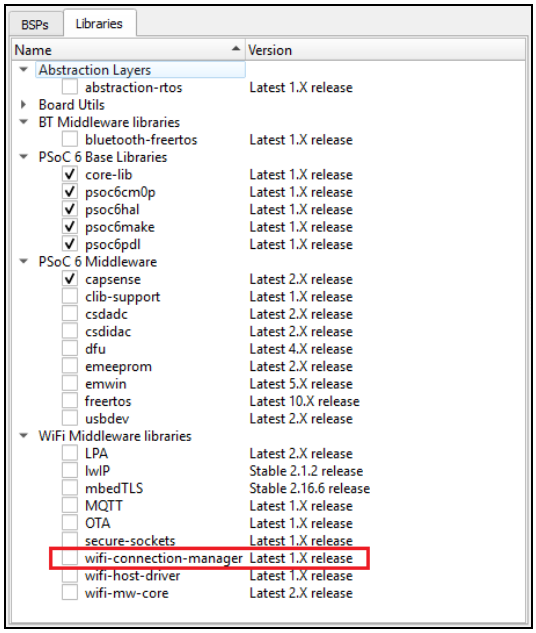
The ModusToolbox Library Manager tool allows you to add and remove board support packages (BSPs) and libraries, as well as select specific versions of BSPs and libraries. Refer to the Library Manager User Guide for more details. As you can see in the following image, the AnyCloud MQTT Client code example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

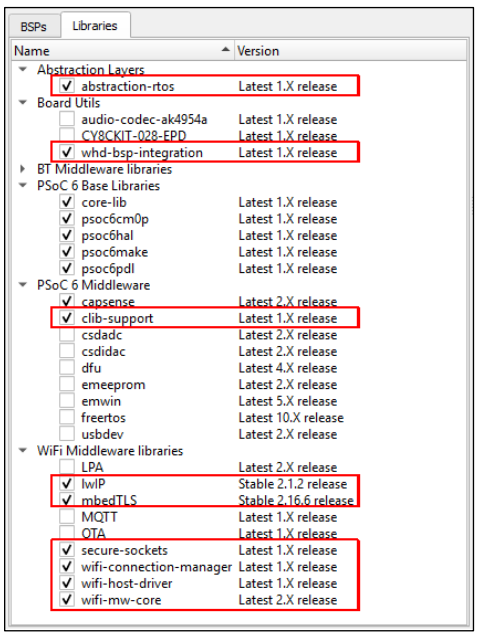


Adding Libraries

As noted earlier, when the ModusToolbox build system encounters a *.lib* file, it adds that library of code to the application. That library may contain additional *.lib* files for dependent libraries. The ModusToolbox build system parses the *.lib* 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 the WCM library.





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

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

Library Reference Tables

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

Note Beginning with the ModusToolbox 2.2 tools release, we've implemented a MTB flow where BSPs and libraries are shared by default. As a result, some new libraries have been created. The old flow, called LIB flow, is still fully supported. For more details about these flows, refer to the ModusToolbox Library Manager Guide.

PSoC 6 Base Libraries

Library Name/Link	Description	Documentation Link
MTB Flow		
core-make	Core Make Build System	API Reference
mtb-hal-cat1	Hardware Abstraction Layer	API Reference
mtb-pdl-cat1	Peripheral Driver Library	API Reference
recipe-make-cat1a	PSoC 6 Make build Recipe	API Reference
LIB Flow		
psoc6hal	Hardware Abstraction Layer	API Reference
psoc6make	GNU make Build System	Repository readme file
psoc6pdl	Peripheral Driver Library	API Reference
Common		
core-lib	Core Library	API Reference
psoc6cm0p	Arm Cortex M0 Plus Prebuilt Image	Repository readme file

PSoC 6 Feature (Middleware) Libraries

Library Name/Link	Description	Documentation Link
capsense	Capacitive Sensing	API Reference
clib-support	C Library Support Functions	
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
freertos	FreeRTOS Kernel	FreeRTOS web page
usbdev	USB Device Library	API Reference

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:

Wi-Fi Middleware Core

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

- *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.

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 (https://github.com/cypresssemiconductorco/wifi-mw-core/optim.html) for specific details about the parameters to adjust.

MQTT

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

- *iot_config.h*: Settings for MQTT.

OTA

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

- *cy_ota_config.h*: Settings for OTA.

Supported Devices

AnyCloud supports the PSoC 62S1 (1 M), PSoC 62S2 (2 M), and PSoC 62S3 (512 k); however, smaller flash devices do not support all use cases or examples. When selecting examples or implementing your application, take the following into account:

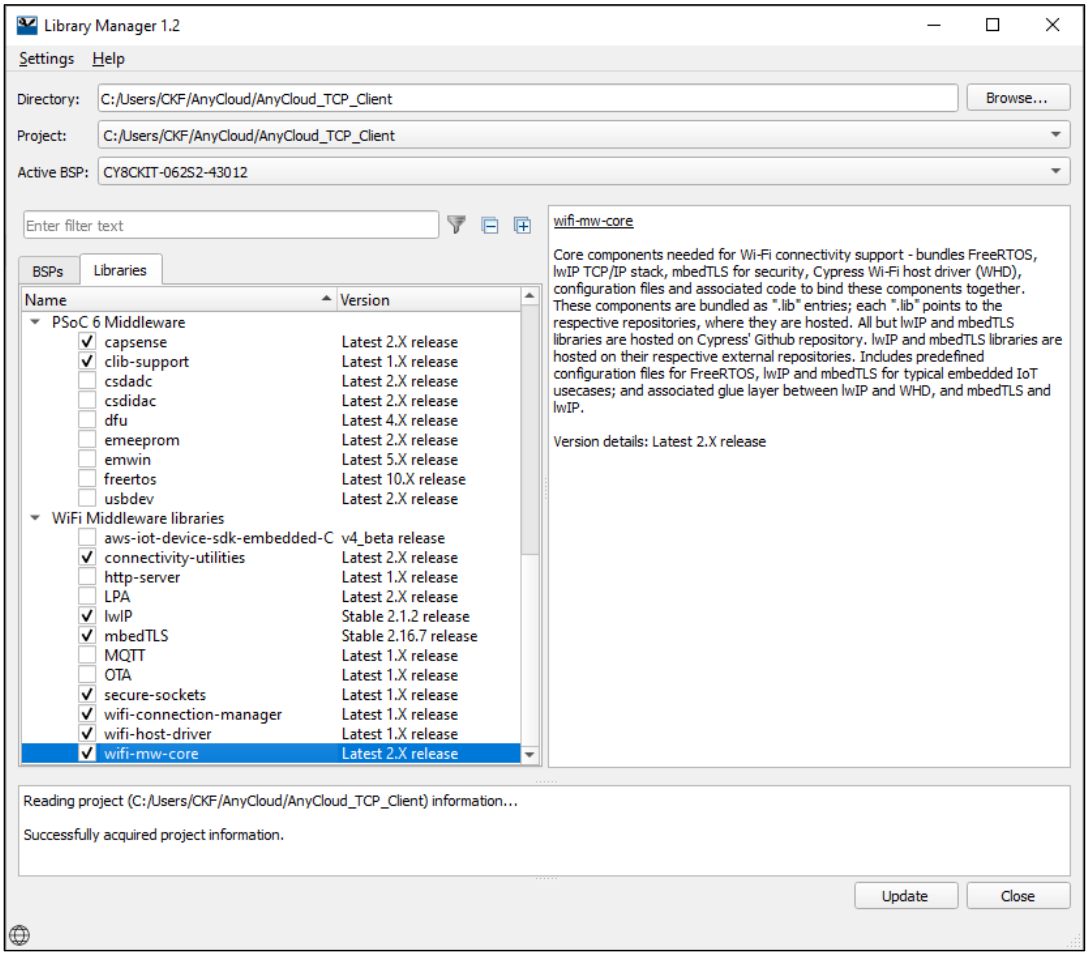
- PSoC 62S1 and PSoC 62S3 do not support OTA. This support will be added in future releases.
- Command Console cannot have all features enabled on PSoC 62S1 and PSoC 62S3 due to low memory footprint. The command framework feature to add application/product use-case-specific commands is supported on these devices. Read the documentation for more information.
- Read about optimizing Mbed TLS and lwIP for your particular use cases in Optimizing Smaller Memory Devices.

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, then create the AnyCloud TCP Client example using your normal process. If you're new to ModusToolbox, use the Eclipse IDE for ModusToolbox. Refer to the Quick Start Guide as needed.

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

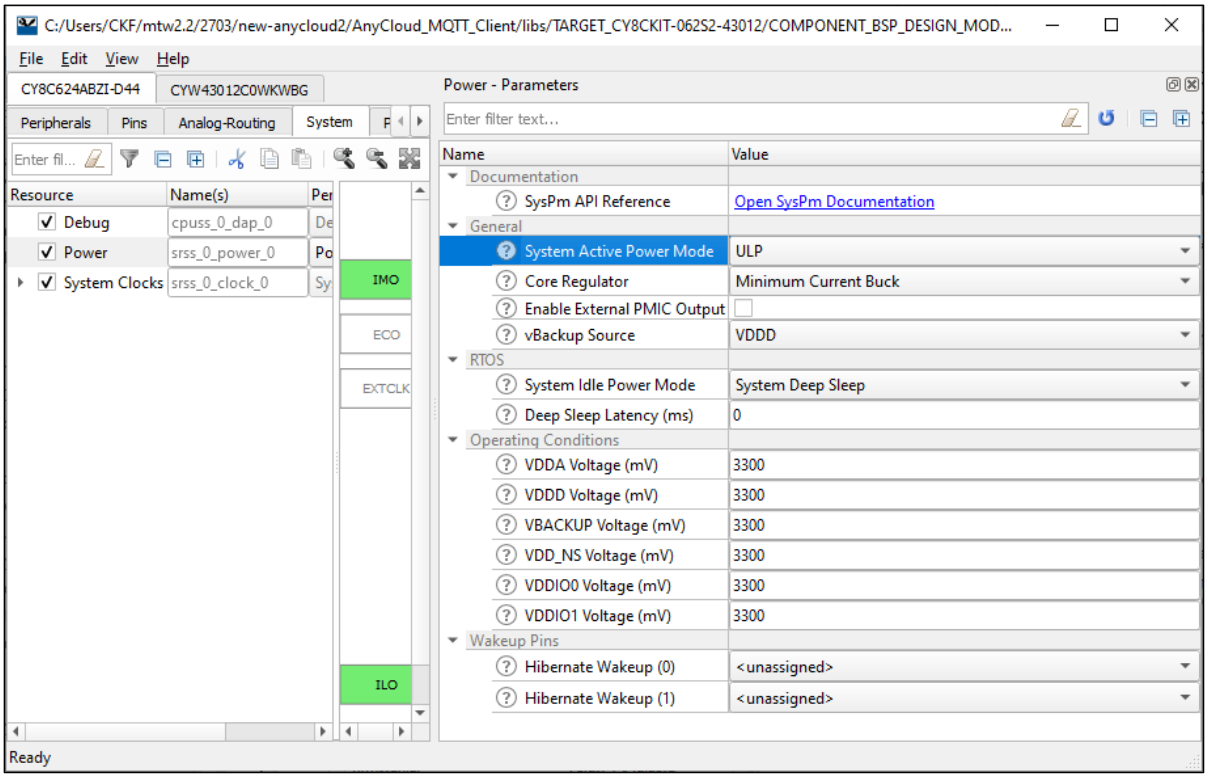


Then, create another application from an example, such as AnyCloud WLAN Low Power. Notice for this application that the LPA library is included.

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.

Adding and Configuring Low Power

The AnyCloud WLAN Low Power example is one of several that demonstrate the low power features in ModusToolbox. 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://cypresssemiconductorco.github.io/lpa/lpa_api_reference_manual/html/index.html

Adding and Configuring Bluetooth

As you have probably already discovered, there are several Bluetooth examples as well. For example, the AnyCloud Wi-Fi Onboarding Using BLE 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.