ModusToolbox Software 2.0
An Overview and Introduction

Dheeraj Kamath - Applications Engineer

WW 20 12

Agenda

- Introduction
- Core Tools
  - What they are and where to get them
  - The make infrastructure and how it works (high level)
  - How you create a project
- Software Enablement
  - BSPs, HALs, PDLs and other libraries
- Supported Ecosystems

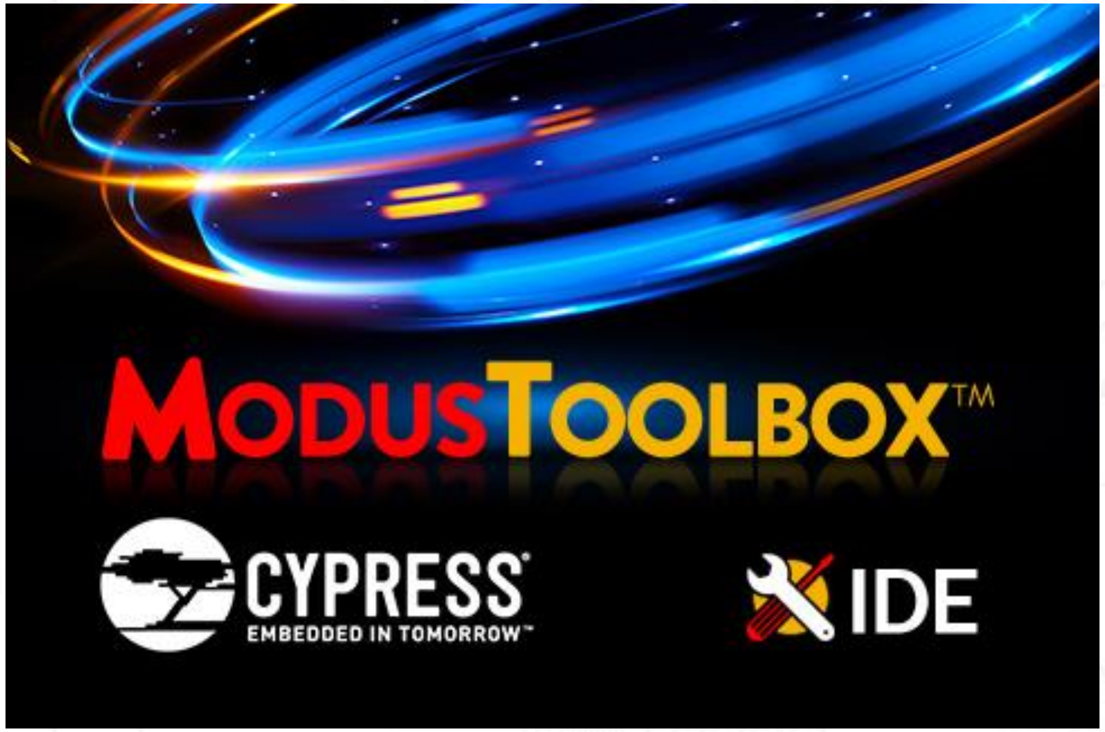
Introduction

What is ModusToolbox Software?

- It is a collection of tools and libraries
- It is compatible across environments (Linux, macOS, Windows)

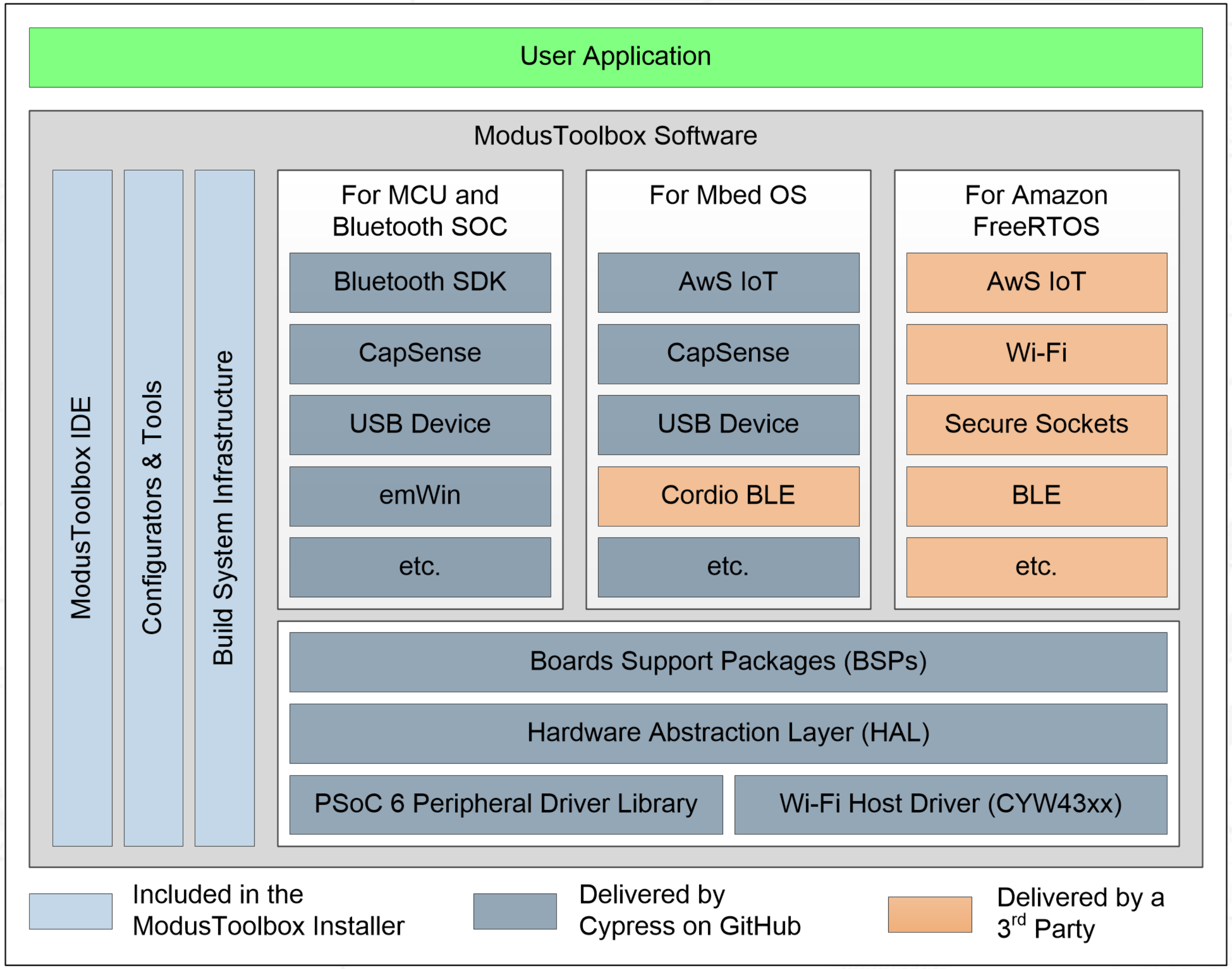
What does it include?

- Configuration tools
- Low-level drivers
- Middleware libraries
- Operating system support
- ModusToolbox IDE



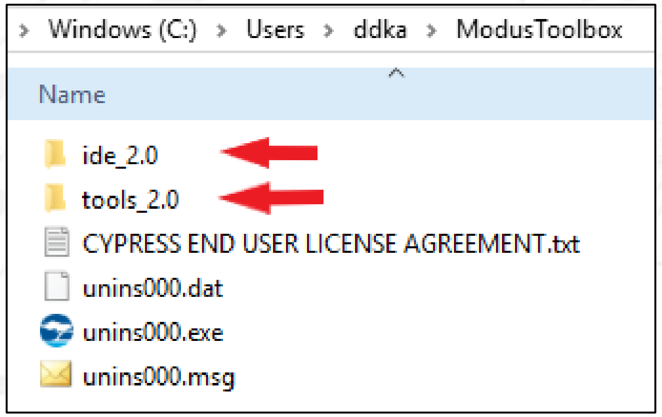
Design Goals

- Comprehensive
  - it has the resources you need
- Flexible
  - you can use the resource in your own workflow
- Atomic
  - you can get just the resources you want


User Application
ModusToolbox Software
ModusToolbox IDE
Configurators & Tools
Build System Infrastructure
For MCU and Bluetooth SOC
Bluetooth SDK
CapSense
USB Device
emWin
etc.
For Mbed OS
AwS IoT
CapSense
USB Device
Cordio BLE
etc.
For Amazon FreeRTOS
AwS IoT
Wi-Fi
Secure Sockets
BLE
etc.
Boards Support Packages (BSPs)
Hardware Abstraction Layer (HAL)
PSoC 6 Peripheral Driver Library
Wi-Fi Host Driver (CYW43xx)
Included in the ModusToolbox Installer
Delivered by Cypress on GitHub
Delivered by a 3rd Party

What is in the MTB Installer?

- Build system infrastructure
- Configurators and tools
- ModusToolbox IDE

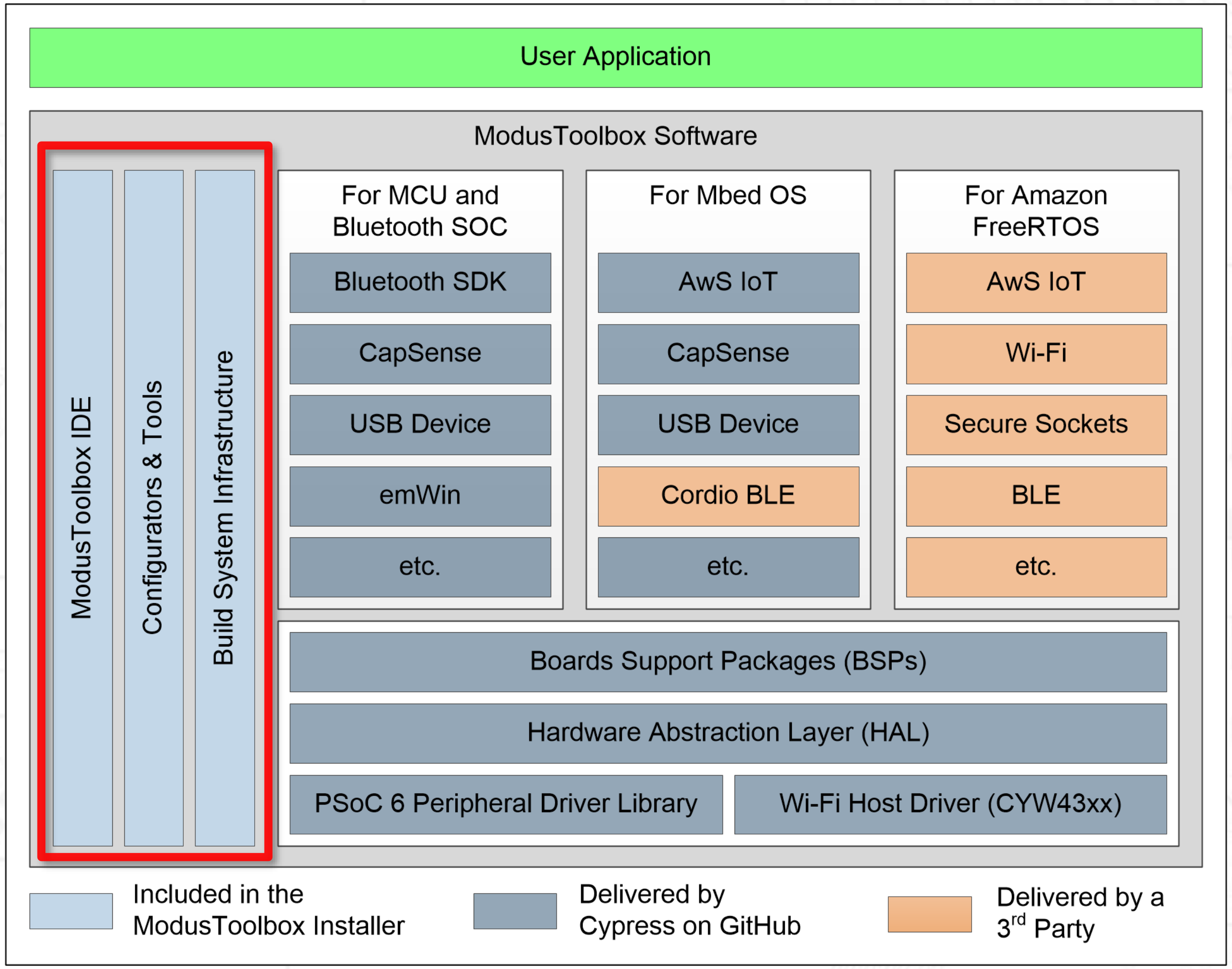


Windows (C:) › Users › ddka › ModusToolbox

Name

- ide_2.0
- tools_2.0
- CYPRESS END USER LICENSE AGREEMENT.txt
- unins000.dat
- unins000.exe
- unins000.msg

<u>Note</u>: Installer does not include enablement software such as driver libraries or middleware



User Application
ModusToolbox Software
ModusToolbox IDE
Configurators & Tools
Build System Infrastructure
For MCU and Bluetooth SOC
Bluetooth SDK
CapSense
USB Device
emWin
etc.
For Mbed OS
AwS IoT
CapSense
USB Device
Cordio BLE
etc.
For Amazon FreeRTOS
AwS IoT
Wi-Fi
Secure Sockets
BLE
etc.
Boards Support Packages (BSPs)
Hardware Abstraction Layer (HAL)
PSoC 6 Peripheral Driver Library
Wi-Fi Host Driver (CYW43xx)
Included in the ModusToolbox Installer
Delivered by Cypress on GitHub
Delivered by a 3rd Party

Where is the SDK?

- It is on GitHub as a collection of independent repositories

  https://github.com/cypresssemiconductorco/

- It also provides libraries to be used in particular ecosystems
  - MCU and Bluetooth SOC ecosystem – PSoC6, Wi-Fi, Bluetooth and BLE
  - Mbed OS ecosystem – Mbed OS, TLS, Cloud Services
  - Amazon FreeRTOS ecosystem – FreeRTOS kernel, AWS libraries, AWS Cloud Services

Core Tools

Core Tools

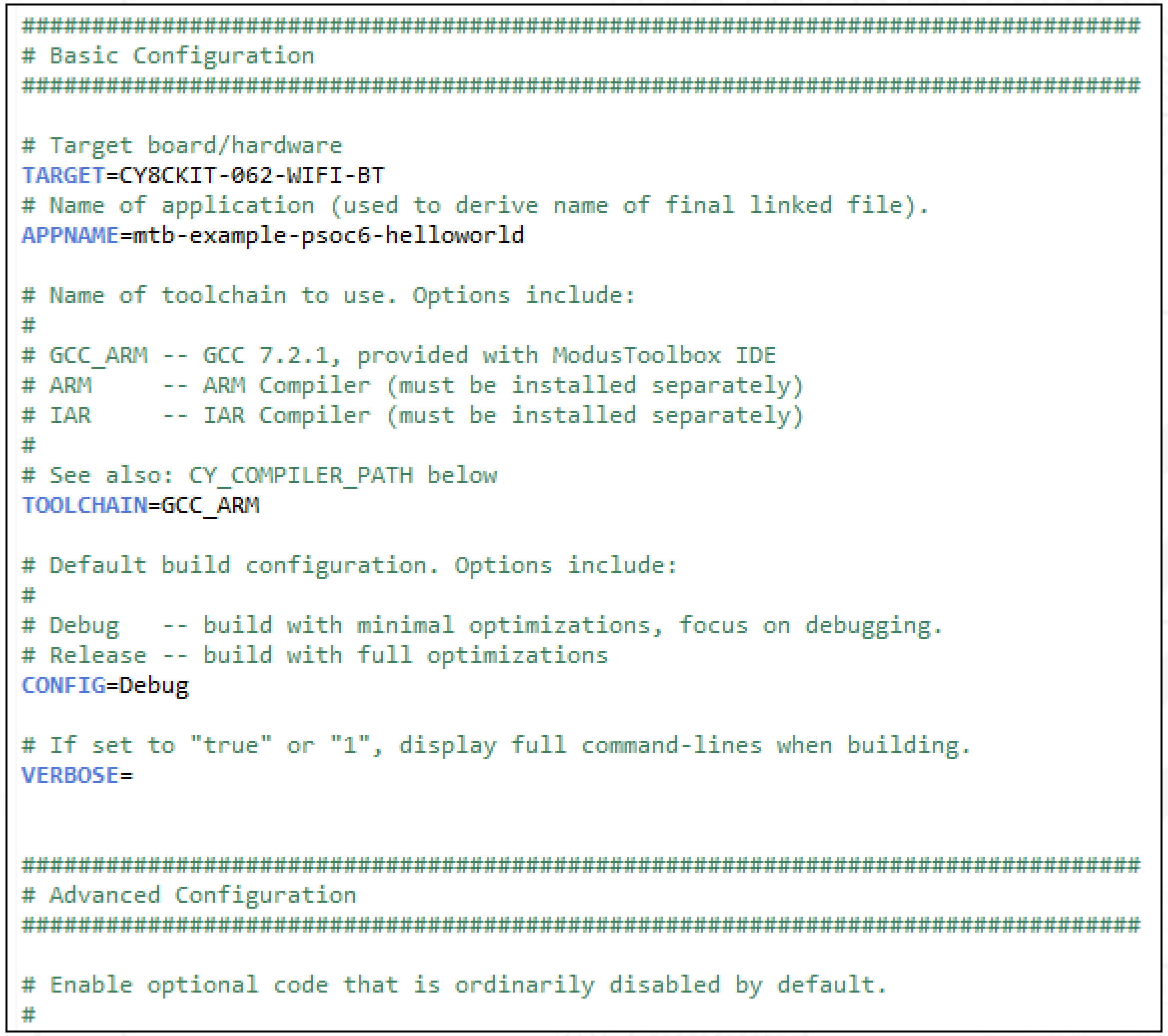
- Build Infrastructure
- Configurators and tools
- Utilities
- ModusToolbox IDE

Build Infrastructure

- What does it do?
  - creates a project
    - either a folder that contains all the files, or a ModusToolbox IDE project
  - creates an executable
  - provides debug capabilities
- A makefile defines everything required for your project, including:
  - the target hardware (board/board support package to use)
  - the libraries to use for the project
  - the build tools to use
  - compiler/assembler/linker flags to control the build
- In MTB 1.1 a similar concept in a file called modus.mk – that is gone
- New makefile is called, get this, makefile ☺

Build Infrastructure

- You set values for variables in the makefile
- You invoke the makefile
- Three ways
  - From the command line
  - Use a new tool called Project Creator
    - just does that, doesn't compile or build
  - Use the ModusToolbox IDE
- Managing the makefile is documented in [KBA229177]()

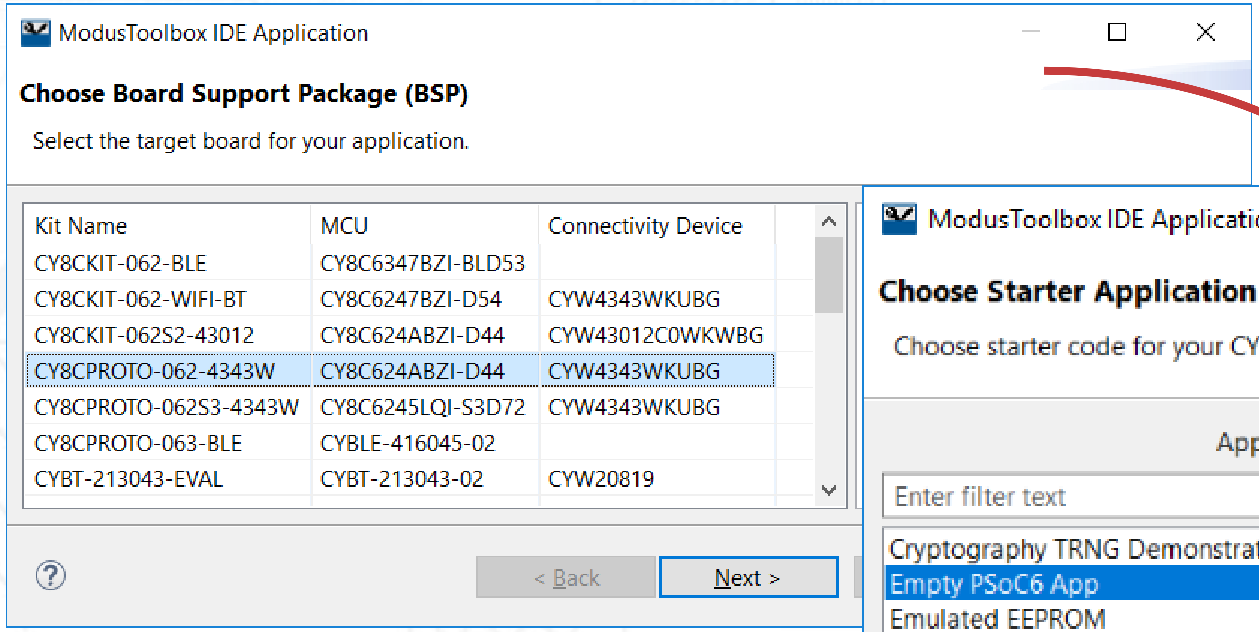


```
################################################################################
# Basic Configuration
################################################################################

# Target board/hardware
TARGET=CY8CKIT-062-WIFI-BT
# Name of application (used to derive name of final linked file).
APPNAME=mtb-example-psoc6-helloworld

# Name of toolchain to use. Options include:
#
# GCC_ARM -- GCC 7.2.1, provided with ModusToolbox IDE
# ARM     -- ARM Compiler (must be installed separately)
# IAR     -- IAR Compiler (must be installed separately)
#
# See also: CY_COMPILER_PATH below
TOOLCHAIN=GCC_ARM

# Default build configuration. Options include:
#
# Debug   -- build with minimal optimizations, focus on debugging.
# Release -- build with full optimizations
CONFIG=Debug

# If set to "true" or "1", display full command-lines when building.
VERBOSE=


################################################################################
# Advanced Configuration
################################################################################

# Enable optional code that is ordinarily disabled by default.
#
```

Create a Project

- In MTB IDE – you get an actual Eclipse-based project
  - Use the New Application wizard
  - No device selection, it is boards only
- Project Creator
  - It is the “New Application” wizard without the IDE
  - You get a project folder with all the files
- Command Line
  - `make getlibs` (covered in upcoming slides)
  - You get a project folder with all the files
- Behind the scenes – all three approaches invoke `make getlibs`
  - That’s what does the real work
  - On the command line it’s explicit, otherwise implicit/hidden

Create a Project (using ModusToolbox IDE)

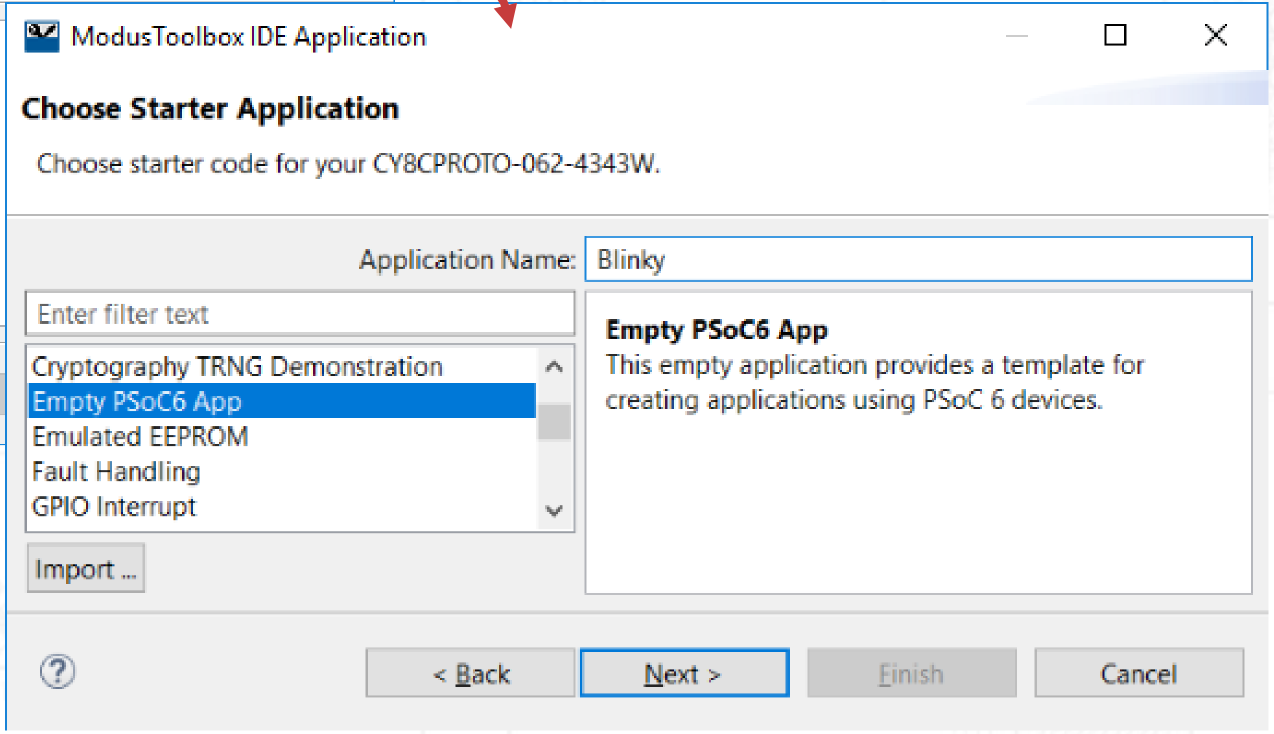


ModusToolbox IDE Application

Choose Board Support Package (BSP)

Select the target board for your application.

Kit Name	MCU	Connectivity Device
CY8CKIT-062-BLE	CY8C6347BZI-BLD53	
CY8CKIT-062-WIFI-BT	CY8C6247BZI-D54	CYW4343WKUBG
CY8CKIT-062S2-43012	CY8C624ABZI-D44	CYW43012C0WKWBG
CY8CPROTO-062-4343W	CY8C624ABZI-D44	CYW4343WKUBG
CY8CPROTO-062S3-4343W	CY8C6245LQI-S3D72	CYW4343WKUBG
CY8CPROTO-063-BLE	CYBLE-416045-02	
CYBT-213043-EVAL	CYBT-213043-02	CYW20819



ModusToolbox IDE Application

Choose Starter Application

Choose starter code for your CY8CPROTO-062-4343W.

Application Name: Blinky

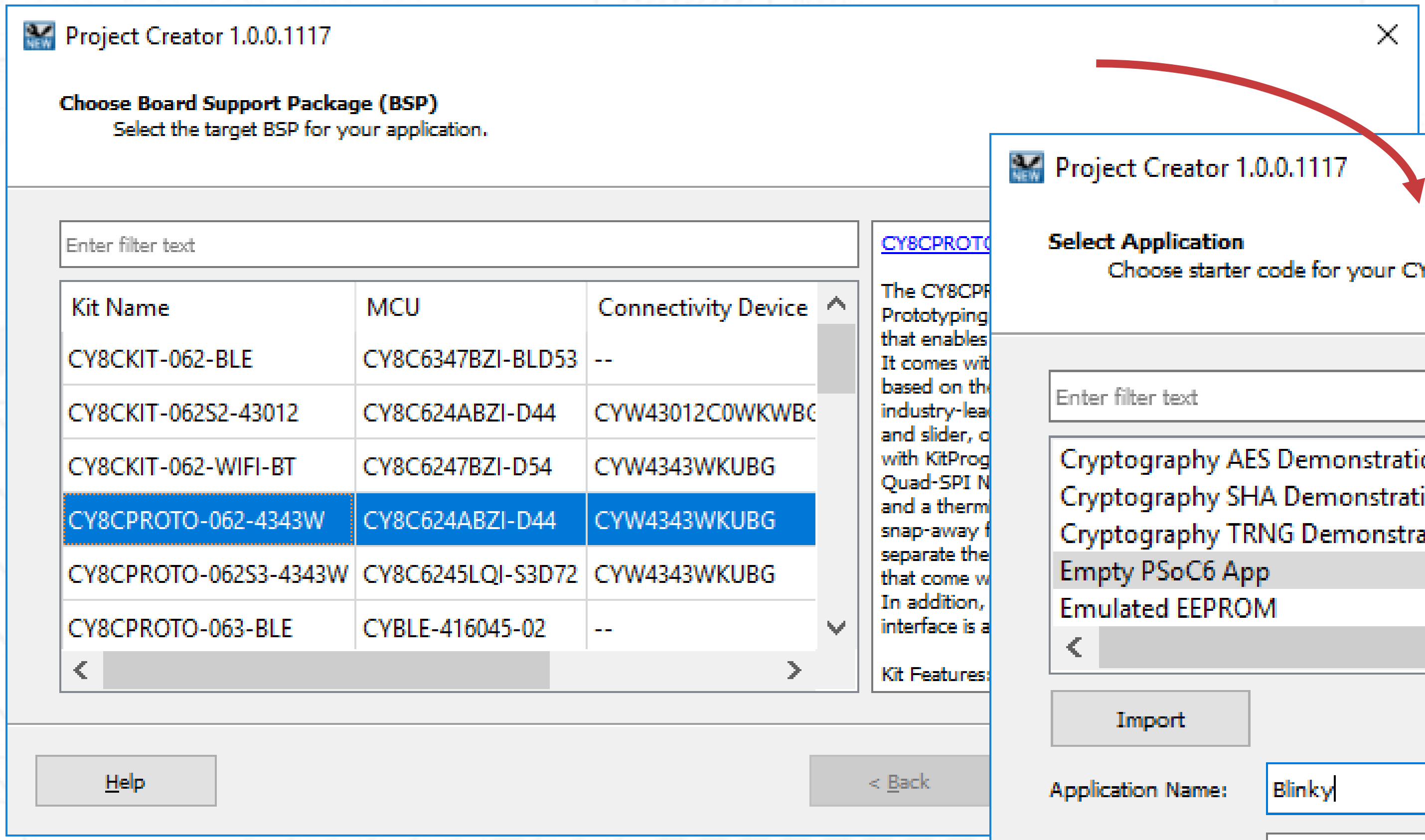
Enter filter text

- Cryptography TRNG Demonstration
- Empty PSoC6 App
- Emulated EEPROM
- Fault Handling
- GPIO Interrupt

Empty PSoC6 App
This empty application provides a template for creating applications using PSoC 6 devices.

Create a Project (using Project Creator)

Tool available under:
"*ModusToolbox\tools_2.0\project-creator*"



Project Creator 1.0.0.1117

Choose Board Support Package (BSP)
Select the target BSP for your application.

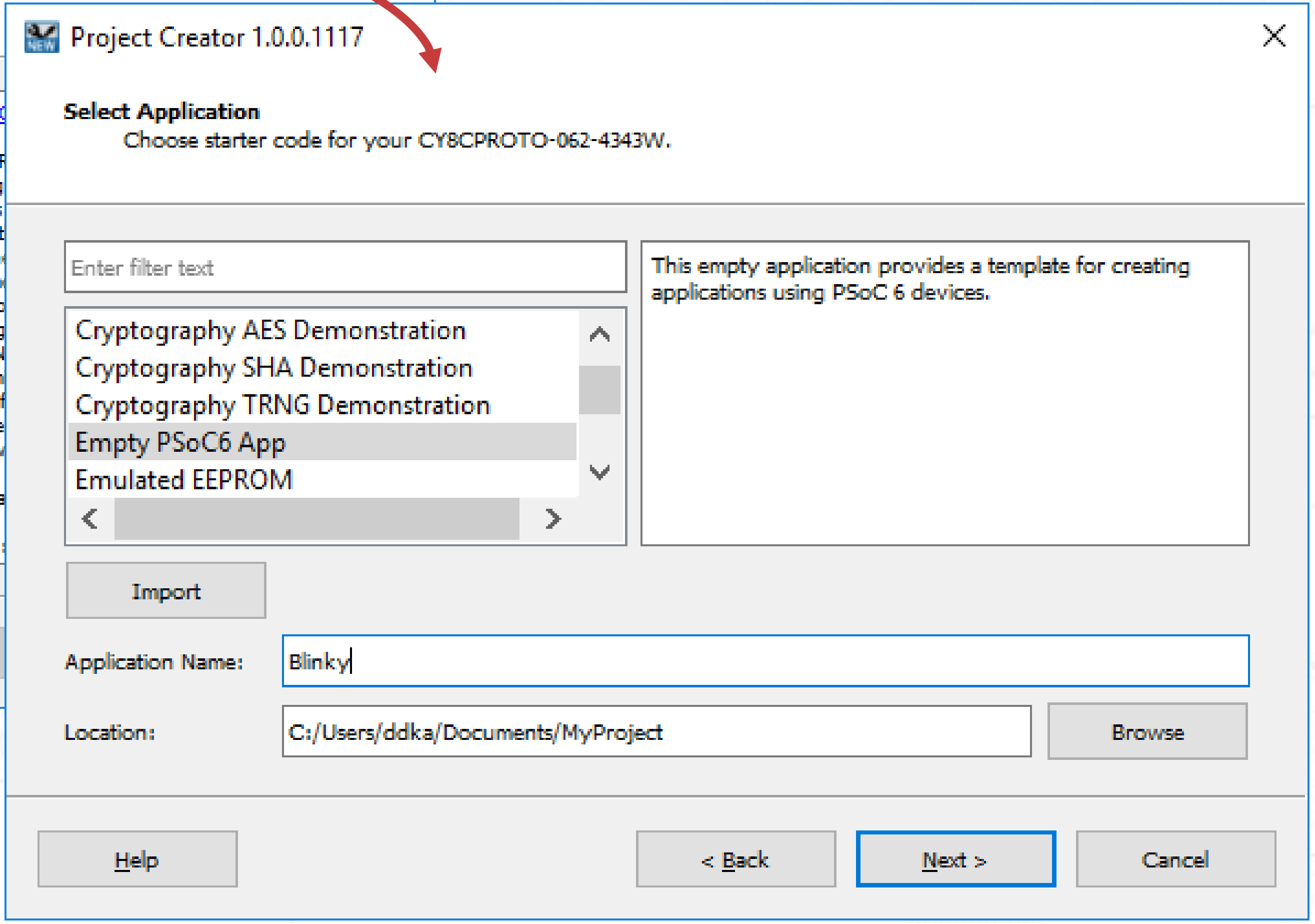
Enter filter text

Kit Name	MCU	Connectivity Device
CY8CKIT-062-BLE	CY8C6347BZI-BLD53	--
CY8CKIT-062S2-43012	CY8C624ABZI-D44	CYW43012C0WKWBG
CY8CKIT-062-WIFI-BT	CY8C6247BZI-D54	CYW4343WKUBG
CY8CPROTO-062-4343W	CY8C624ABZI-D44	CYW4343WKUBG
CY8CPROTO-062S3-4343W	CY8C6245LQI-S3D72	CYW4343WKUBG
CY8CPROTO-063-BLE	CYBLE-416045-02	--

CY8CPROTO

The CY8CPR
Prototyping
that enables
It comes wit
based on th
industry-lea
and slider, c
with KitProg
Quad-SPI N
and a therm
snap-away f
separate the
that come w
In addition,
interface is a

Kit Features:



Project Creator 1.0.0.1117

Select Application
Choose starter code for your CY8CPROTO-062-4343W.

Enter filter text

Cryptography AES Demonstration
Cryptography SHA Demonstration
Cryptography TRNG Demonstration
Empty PSoC6 App
Emulated EEPROM

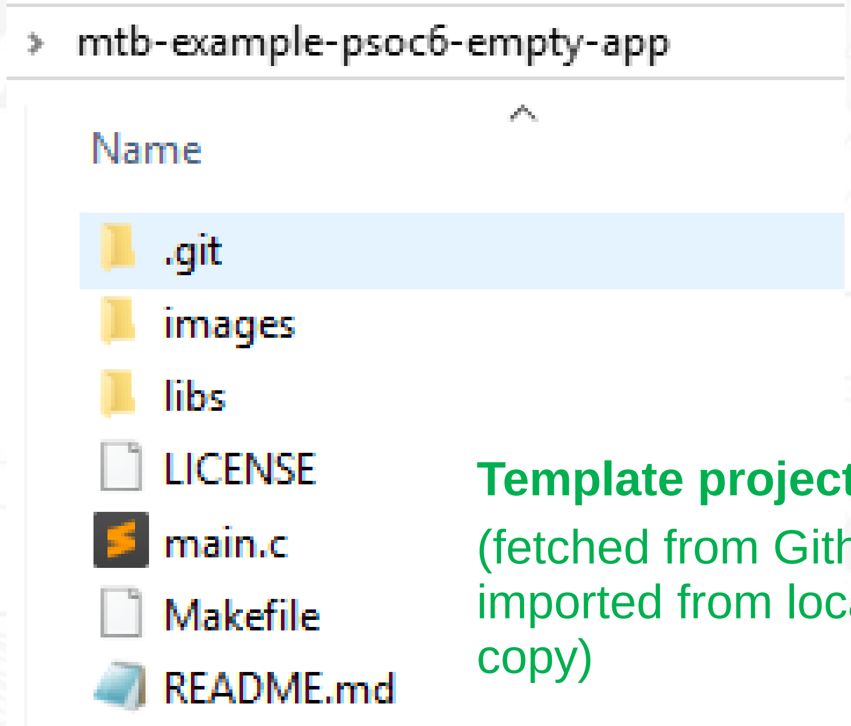
This empty application provides a template for creating applications using PSoC 6 devices.

Application Name: Blinky

Location: C:/Users/ddka/Documents/MyProject

Autodiscover and Project Creation

- When creating a project, it starts in the folder containing the makefile
- It creates a new folder (based on project name) and copies all required resources into the project folder



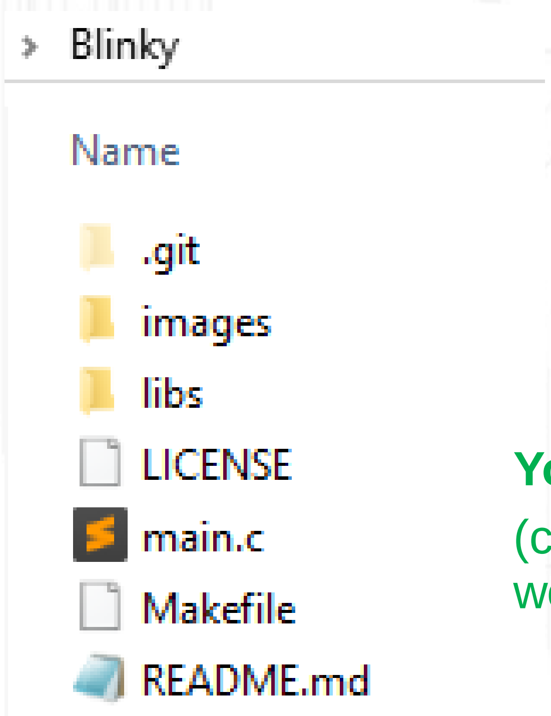
› mtb-example-psoc6-empty-app

Name

.git
images
libs
LICENSE
main.c
Makefile
README.md

Template project

(fetched from Github / imported from local copy)



› Blinky

Name

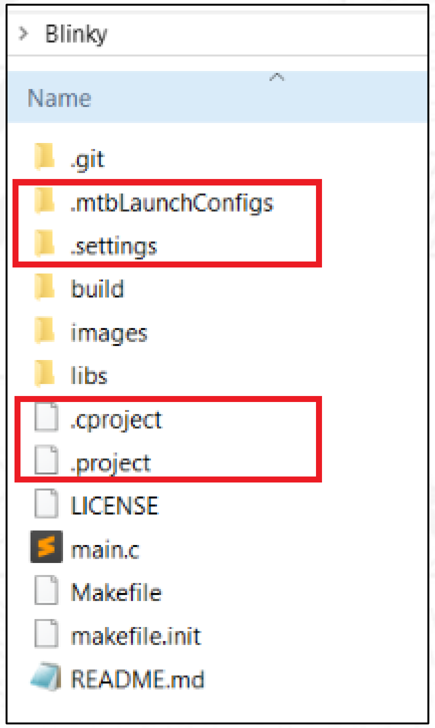
.git
images
libs
LICENSE
main.c
Makefile
README.md

Your project

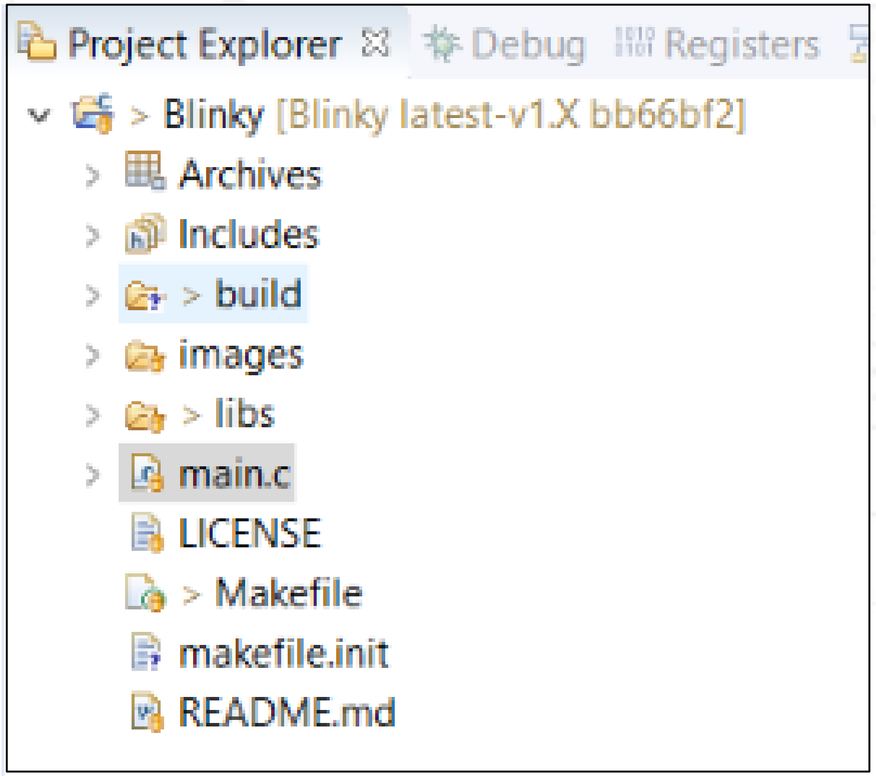
(created in your workspace directory**)**

Autodiscover and Project Creation

- All files appear in a project folder
  - Using Project Creator, wherever you specify
  - Using the IDE, in the Eclipse workspace
- If you use the IDE, you also get a project file



```
> Blinky
Name
.git
.mtbLaunchConfigs
.settings
build
images
libs
.cproject
.project
LICENSE
main.c
Makefile
makefile.init
README.md
```



```
> Blinky [Blinky latest-v1.X bb66bf2]
    Archives
    Includes
    > build
    images
    > libs
    main.c
    LICENSE
    > Makefile
    makefile.init
    README.md
```

Autodiscover and Project Creation

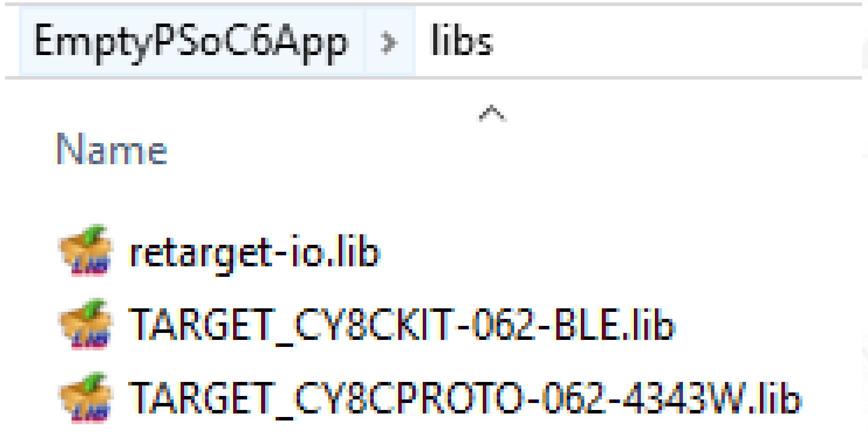
- Build system automatically discovers all .c, .h, .cpp, .s, .a, .o files in the makefile tree
  - They are replicated in the project folder
- You can add files that reside outside the makefile folder
  - add another directory using the CY_SHAREDLIB_PATH variable
  - You can explicitly list files in the SOURCES and INCLUDES make variables

Example:

```
INCLUDES=../MySource1 ../MySource2
SOURCES=$(wildcard ../MySource1/*.c) $(wildcard ../MySource2/*.c)
```

Libraries, .lib Files, and `make getlibs`

- Add a library (e.g. middleware) by using a .lib file
- You create a .lib file for each library you want to use – this is what's in the file
  - https://github.com/cypresssemiconductorco/emwin/#release-v5.48.1
  - The last part of the URL is a git tag that specifies a particular commit I want to use
  - A release tag never moves, so you can point your app to a stable, known-good release
  - Instead of a tag, you can use an actual commit hash
- `make getlibs`
  - Finds each .lib file
  - Reads it
  - Clones the repository
  - Checks out the code for the specific commit
  - Adds those files to the *libs* folder in the new project folder



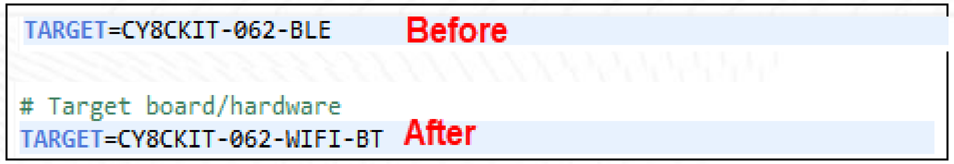
EmptyPSoC6App › libs

Name

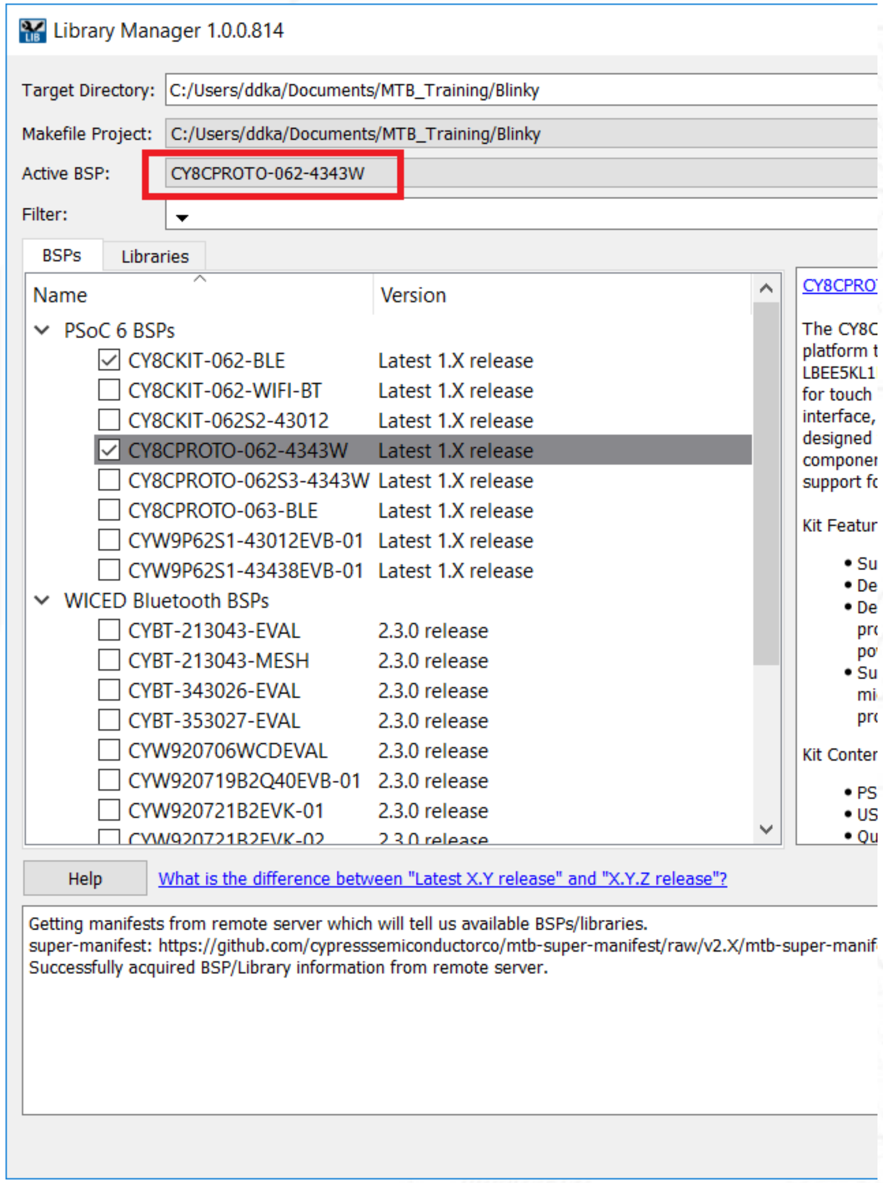
retarget-io.lib
TARGET_CY8CKIT-062-BLE.lib
TARGET_CY8CPROTO-062-4343W.lib

Library Manager

- Stand alone tool or inside the IDE
- Add, remove, change version of any library
  - You can pick a BSP!
- To retarget an application to a new kit
  - Open the Library manager
  - Pick the new kit
- The new kit must support the same features/functionality but…
  - All BSPs use a consistent set of definitions
- Library manager updates the makefile



```
TARGET=CY8CKIT-062-BLE        Before

# Target board/hardware
TARGET=CY8CKIT-062-WIFI-BT  After
```



Library Manager 1.0.0.814

Target Directory: C:/Users/ddka/Documents/MTB_Training/Blinky
Makefile Project: C:/Users/ddka/Documents/MTB_Training/Blinky
Active BSP: CY8CPROTO-062-4343W
Filter:

Name	Version
PSoC 6 BSPs	
✓ CY8CKIT-062-BLE	Latest 1.X release
☐ CY8CKIT-062-WIFI-BT	Latest 1.X release
☐ CY8CKIT-062S2-43012	Latest 1.X release
✓ CY8CPROTO-062-4343W	Latest 1.X release
☐ CY8CPROTO-062S3-4343W	Latest 1.X release
☐ CY8CPROTO-063-BLE	Latest 1.X release
☐ CYW9P62S1-43012EVB-01	Latest 1.X release
☐ CYW9P62S1-43438EVB-01	Latest 1.X release
WICED Bluetooth BSPs	
☐ CYBT-213043-EVAL	2.3.0 release
☐ CYBT-213043-MESH	2.3.0 release
☐ CYBT-343026-EVAL	2.3.0 release
☐ CYBT-353027-EVAL	2.3.0 release
☐ CYW920706WCDEVAL	2.3.0 release
☐ CYW920719B2Q40EVB-01	2.3.0 release
☐ CYW920721B2EVK-01	2.3.0 release
☐ CYW920721B2EVK-02	2.3.0 release

CY8CPRO

The CY8C
platform t
LBEE5KL1
for touch
interface,
designed
componer
support fc

Kit Featur
- Su
- De
- De
  pr
  po
- Su
  mi
  pr

Kit Conter
- PS
- US
- Qu

What is the difference between "Latest X.Y release" and "X.Y.Z release"?

```
Getting manifests from remote server which will tell us available BSPs/libraries.
super-manifest: https://github.com/cypresssemiconductorco/mtb-super-manifest/raw/v2.X/mtb-super-manif
Successfully acquired BSP/Library information from remote server.
```

Create an Executable

- Build infrastructure provides several variables you use to control the build
- MTB Installs GNU Arm toolchain, that's the default

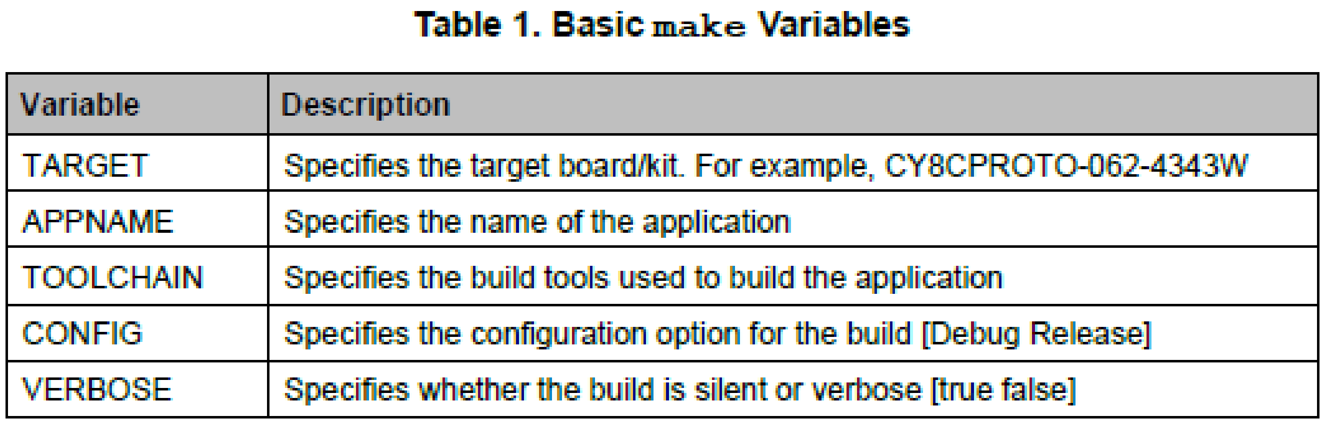
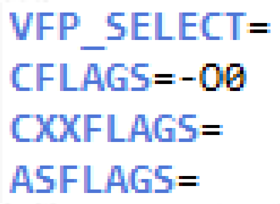


Table 1. Basic `make` Variables

Variable	Description
TARGET	Specifies the target board/kit. For example, CY8CPROTO-062-4343W
APPNAME	Specifies the name of the application
TOOLCHAIN	Specifies the build tools used to build the application
CONFIG	Specifies the configuration option for the build [Debug Release]
VERBOSE	Specifies whether the build is silent or verbose [true false]



```
VFP_SELECT=
CFLAGS=-O0
CXXFLAGS=
ASFLAGS=
```

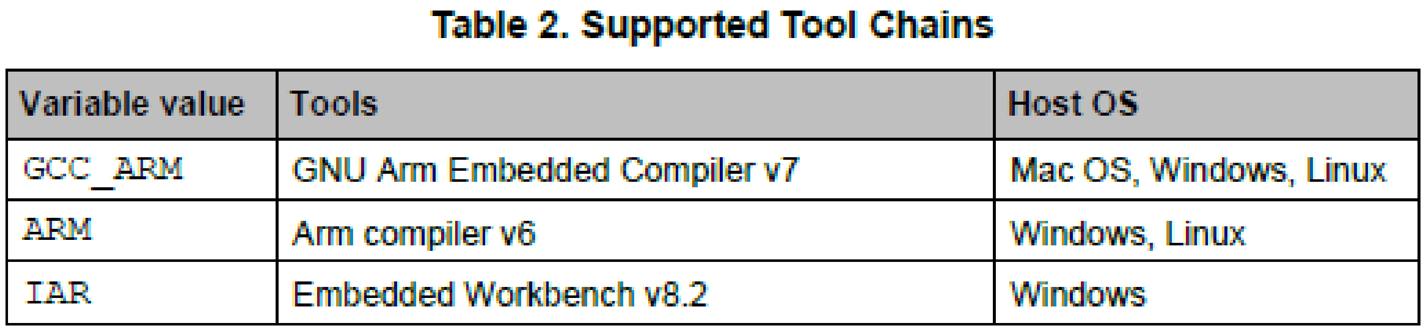
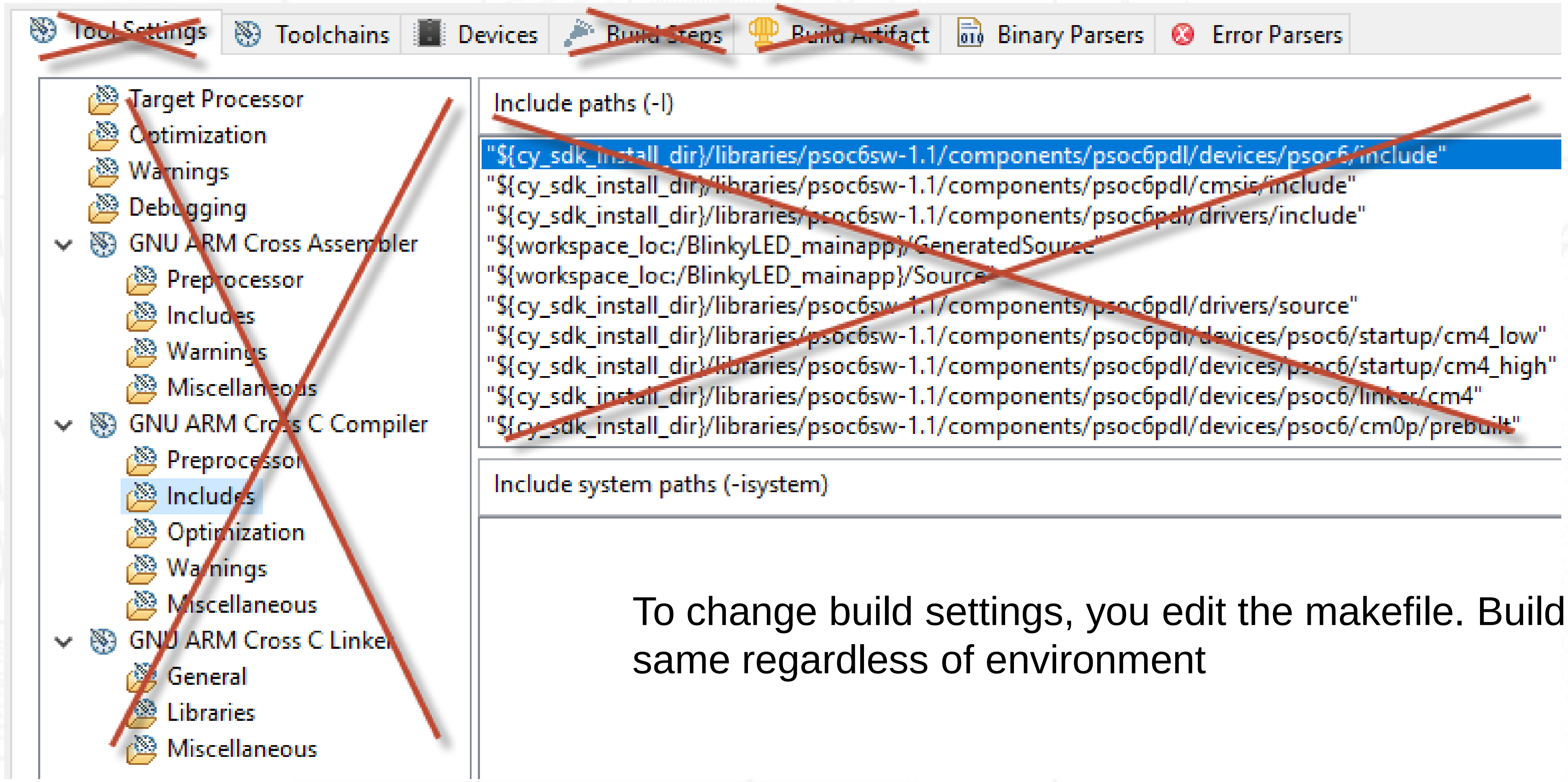


Table 2. Supported Tool Chains

Variable value	Tools	Host OS
GCC_ARM	GNU Arm Embedded Compiler v7	Mac OS, Windows, Linux
ARM	Arm compiler v6	Windows, Linux
IAR	Embedded Workbench v8.2	Windows

Building Code from Within the IDE

- A build command in the IDE wraps around `make build` from the command line
- The “standard” Eclipse compiler build settings are all gone



~~Tool Settings~~ | Toolchains | Devices | ~~Build Steps~~ | ~~Build Artifact~~ | Binary Parsers | Error Parsers

- Target Processor
- Optimization
- Warnings
- Debugging
- GNU ARM Cross Assembler
  - Preprocessor
  - Includes
  - Warnings
  - Miscellaneous
- GNU ARM Cross C Compiler
  - Preprocessor
  - Includes
  - Optimization
  - Warnings
  - Miscellaneous
- GNU ARM Cross C Linker
  - General
  - Libraries
  - Miscellaneous

Include paths (-I)

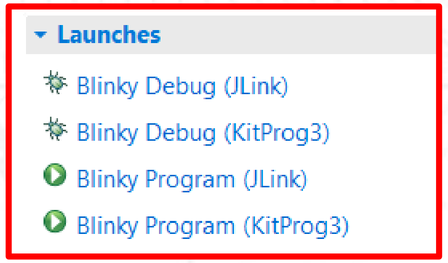
```
"${cy_sdk_install_dir}/libraries/psoc6sw-1.1/components/psoc6pdl/devices/psoc6/include"
"${cy_sdk_install_dir}/libraries/psoc6sw-1.1/components/psoc6pdl/cmsis/include"
"${cy_sdk_install_dir}/libraries/psoc6sw-1.1/components/psoc6pdl/drivers/include"
"${workspace_loc:/BlinkyLED_mainapp}/GeneratedSource"
"${workspace_loc:/BlinkyLED_mainapp}/Source"
"${cy_sdk_install_dir}/libraries/psoc6sw-1.1/components/psoc6pdl/drivers/source"
"${cy_sdk_install_dir}/libraries/psoc6sw-1.1/components/psoc6pdl/devices/psoc6/startup/cm4_low"
"${cy_sdk_install_dir}/libraries/psoc6sw-1.1/components/psoc6pdl/devices/psoc6/startup/cm4_high"
"${cy_sdk_install_dir}/libraries/psoc6sw-1.1/components/psoc6pdl/devices/psoc6/linker/cm4"
"${cy_sdk_install_dir}/libraries/psoc6sw-1.1/components/psoc6pdl/devices/psoc6/cm0p/prebuilt"
```

Include system paths (-isystem)

To change build settings, you edit the makefile. Build behaves precisely the same regardless of environment

Program and Debug

- the application creation process generates launch configurations



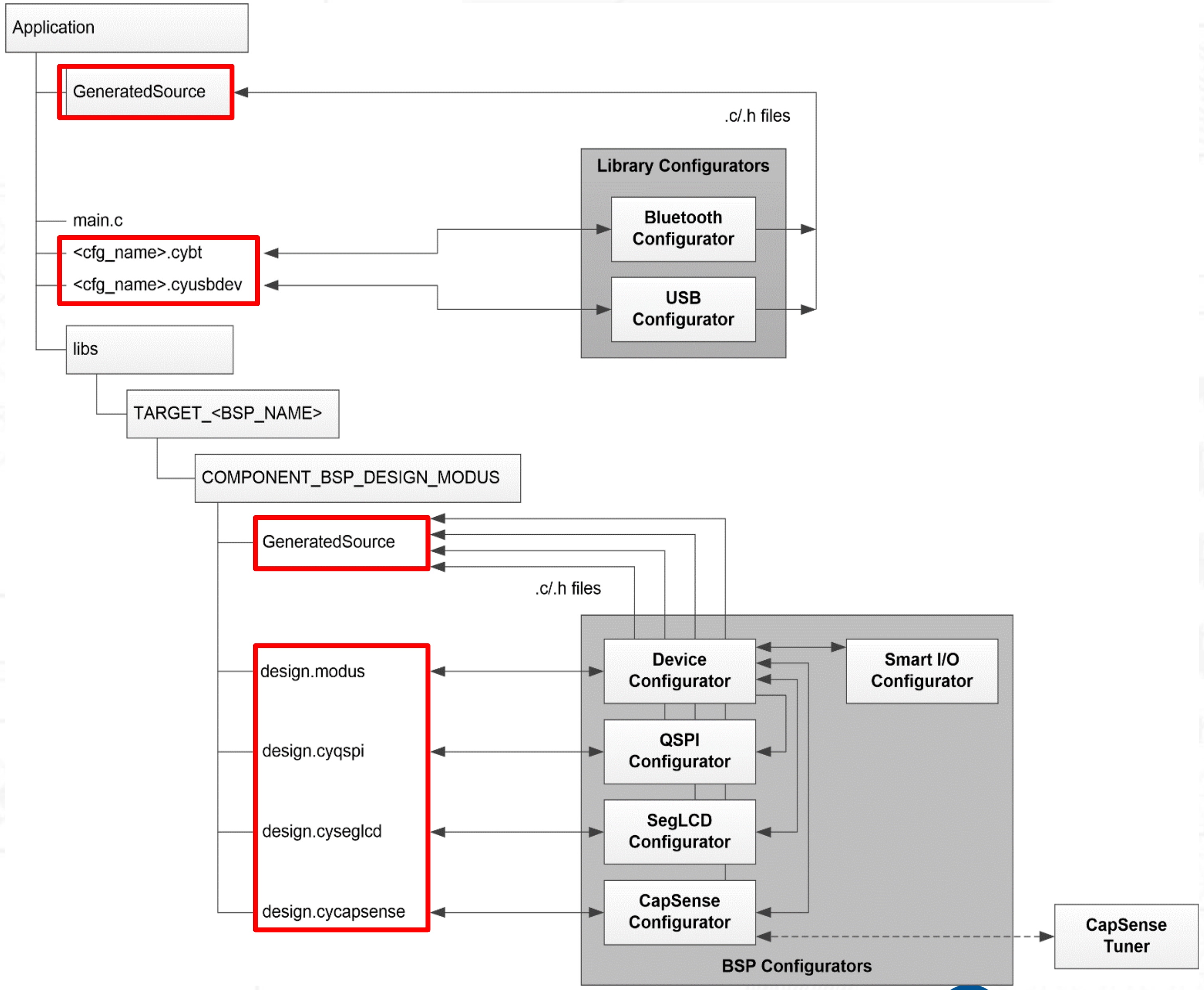
Launches
- Blinky Debug (JLink)
- Blinky Debug (KitProg3)
- Blinky Program (JLink)
- Blinky Program (KitProg3)

Table 3. ModusToolbox Program & Debug Tools

Tool	Description	Documentation
Cypress Programmer	Cypress Programmer functionality is built into ModusToolbox Software. Cypress Programmer is also available as a standalone tool.	Programming Tools page, go to the documentation tab
fw-loader	A simple command line tool to identify which version of KitProg is on a Cypress kit, and easily switch back and forth between legacy KitProg2 and current KitProg3.	*readme.txt* file in the tool folder
KitProg3	This tool is managed by fw-loader, it is not available separately. KitProg3 is Cypress' low-level communication/debug firmware that supports CMSIS-DAP and DAPLink (for Mbed OS). Use fw-loader to upgrade your kit to KitProg3, if it has KitProg2 installed.	User Guide
OpenOCD	A Cypress-specific implementation of OpenOCD is installed with ModusToolbox software.	Developer's Guide
DAPLink	Support is implemented through KitProg3	DAPLink Handbook

Configurators and Tools

- Introduces the concept of BSP Configurator and Library Configurator
- Some changes in storing data
  - It is not all in the *design.modus* file
- There are two *GeneratedSource* folders in different places

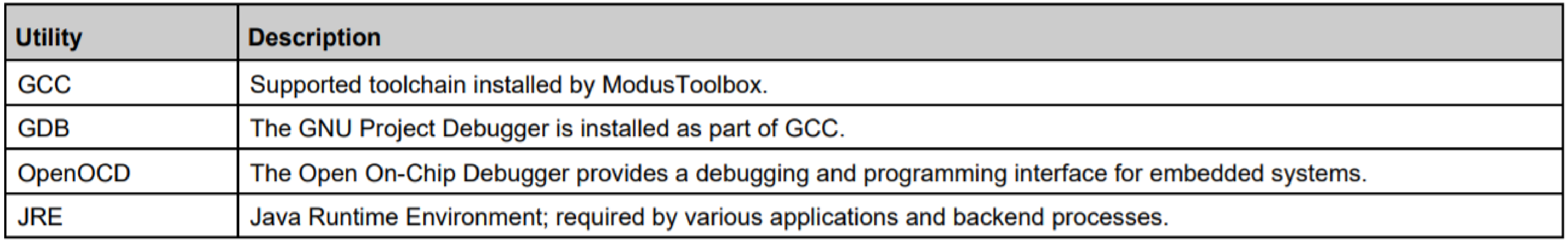


New Configurators and Tools

Configurator or Tool	Description
seglcd-configurator	Configure a generic LCD Direct Segment Drive controller for a variety of LCD glass at different voltage levels with multiplex ratios up to 16x.
project-creator	Create a new application project. This tool is a stand-alone version of the IDE wizard, available as a GUI and a command-line tool (CLI).
cype-tool	The power estimator tool provides an estimate of the power consumed by a target device.
library-manager	Select which library (including BSPs) and which version of each you want to include. Updates the makefile based on your choices.

Utilities

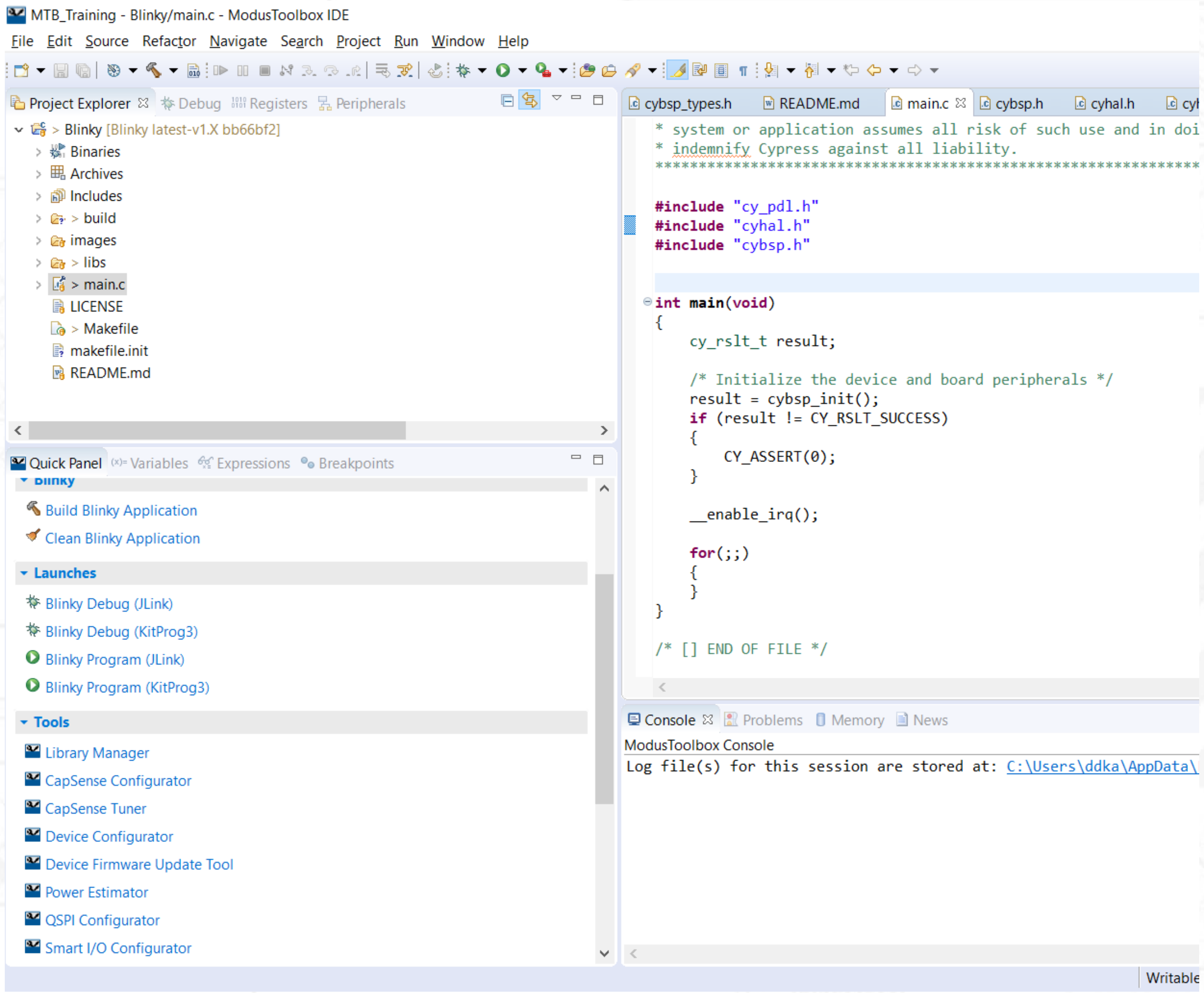
- Commonly used utilities that are required for application development



Utility	Description
GCC	Supported toolchain installed by ModusToolbox.
GDB	The GNU Project Debugger is installed as part of GCC.
OpenOCD	The Open On-Chip Debugger provides a debugging and programming interface for embedded systems.
JRE	Java Runtime Environment; required by various applications and backend processes.

ModusToolbox IDE

- 100% Optional
- The command line infrastructure behind this makes that so
- For the IDE, the make infrastructure also creates debug configurations



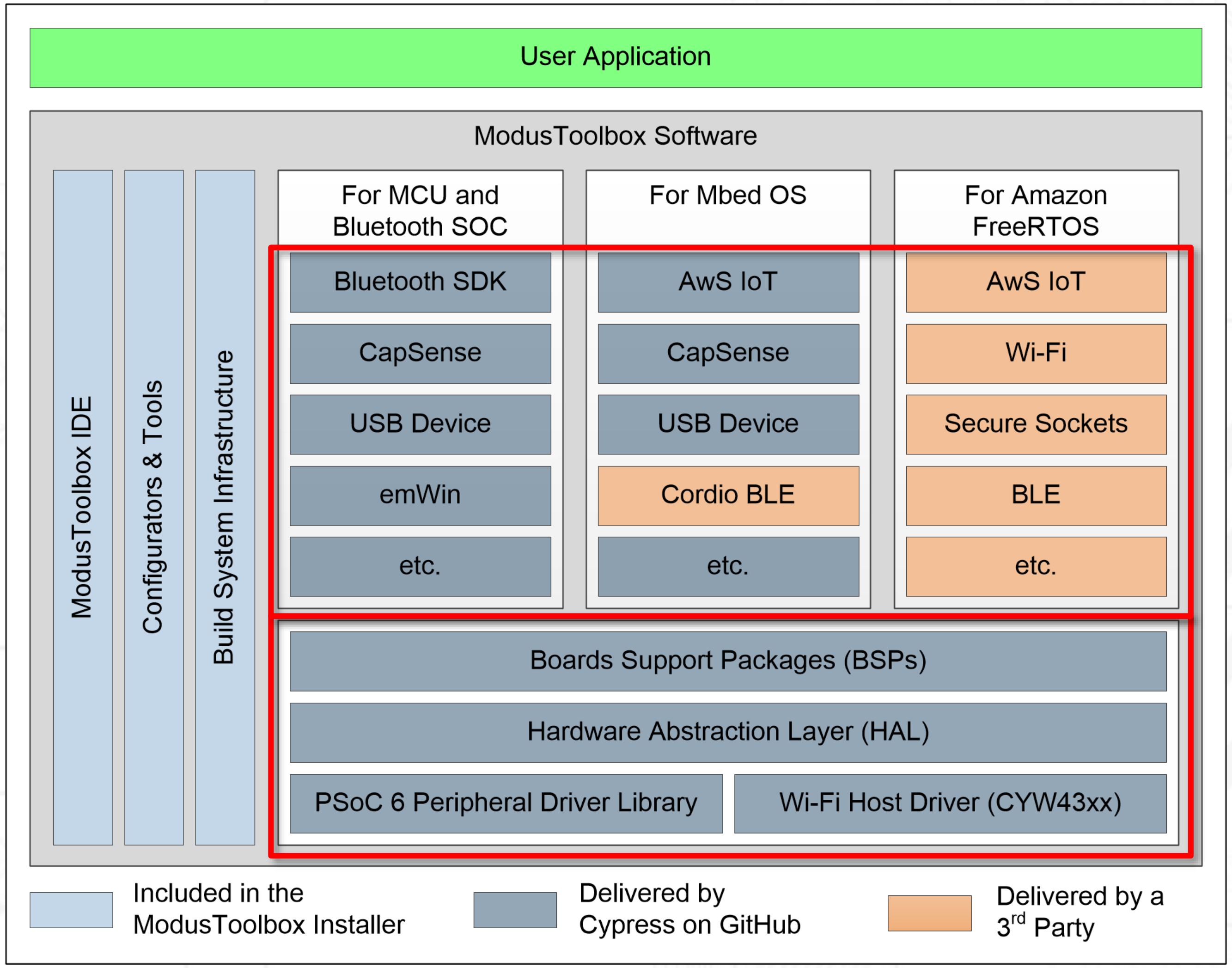
```
 * system or application assumes all risk of such use and in doi
 * indemnify Cypress against all liability.
***************************************************************

#include "cy_pdl.h"
#include "cyhal.h"
#include "cybsp.h"


int main(void)
{
    cy_rslt_t result;

    /* Initialize the device and board peripherals */
    result = cybsp_init();
    if (result != CY_RSLT_SUCCESS)
    {
        CY_ASSERT(0);
    }

    __enable_irq();

    for(;;)
    {
    }
}

/* [] END OF FILE */
```

```
ModusToolbox Console
Log file(s) for this session are stored at: C:\Users\ddka\AppData\
```

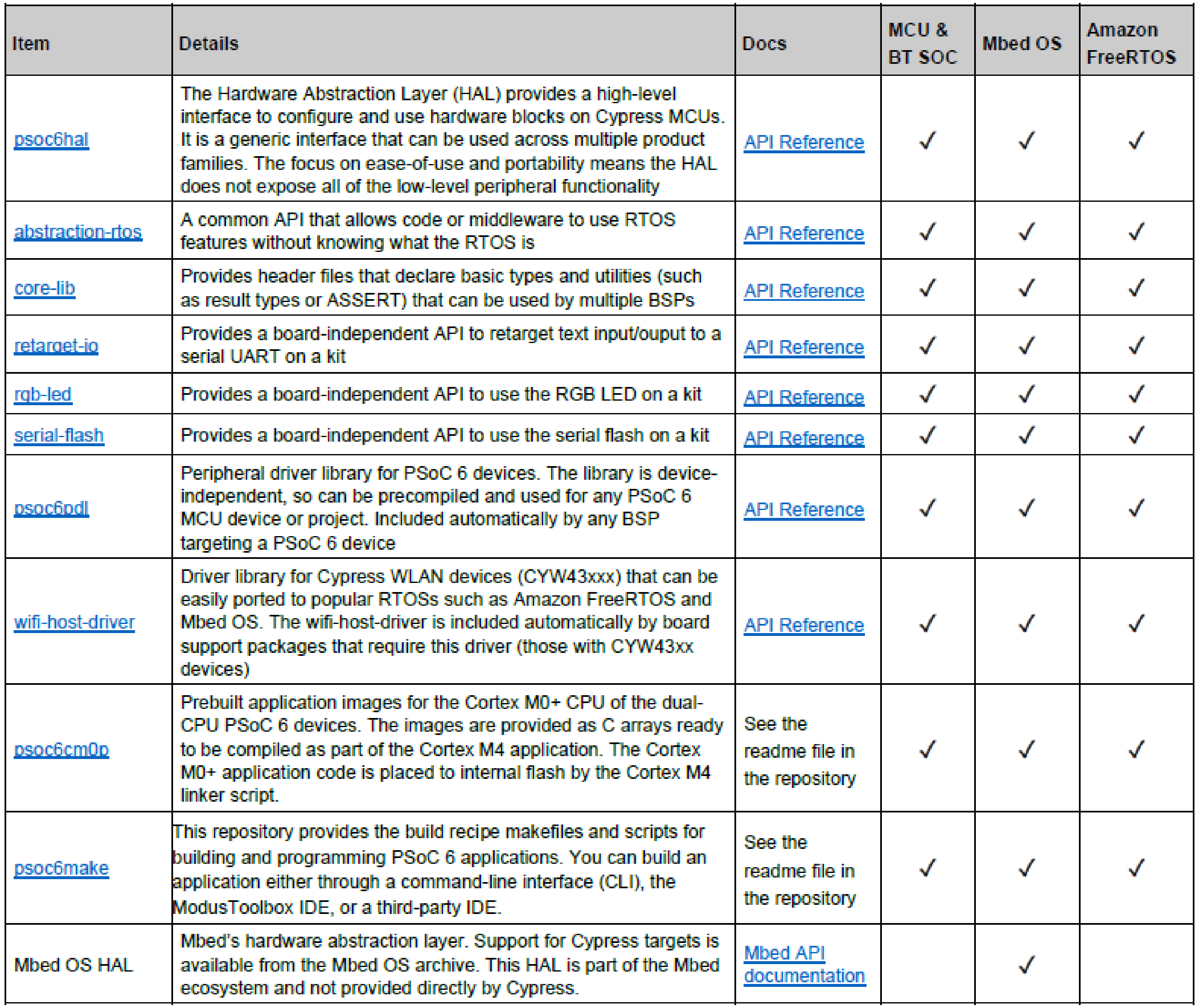
Software Enablement

What Have We Got?

- Aimed at three key ecosystems
  - PSoC 6 MCU and Bluetooth SoC ecosystem development
  - Mbed OS ecosystem development
  - Amazon FreeRTOS ecosystem development
- These broad categories
  - Low-level resources
  - Middleware
  - BSPs
  - Code examples


User Application
ModusToolbox Software
ModusToolbox IDE
Configurators & Tools
Build System Infrastructure
For MCU and Bluetooth SOC
For Mbed OS
For Amazon FreeRTOS
Bluetooth SDK
CapSense
USB Device
emWin
etc.
AwS IoT
CapSense
USB Device
Cordio BLE
etc.
AwS IoT
Wi-Fi
Secure Sockets
BLE
etc.
Boards Support Packages (BSPs)
Hardware Abstraction Layer (HAL)
PSoC 6 Peripheral Driver Library
Wi-Fi Host Driver (CYW43xx)
Included in the ModusToolbox Installer
Delivered by Cypress on GitHub
Delivered by a 3rd Party

Low-level Resources



Item	Details	Docs	MCU & BT SOC	Mbed OS	Amazon FreeRTOS
psoc6hal	The Hardware Abstraction Layer (HAL) provides a high-level interface to configure and use hardware blocks on Cypress MCUs. It is a generic interface that can be used across multiple product families. The focus on ease-of-use and portability means the HAL does not expose all of the low-level peripheral functionality	API Reference	✓	✓	✓
abstraction-rtos	A common API that allows code or middleware to use RTOS features without knowing what the RTOS is	API Reference	✓	✓	✓
core-lib	Provides header files that declare basic types and utilities (such as result types or ASSERT) that can be used by multiple BSPs	API Reference	✓	✓	✓
retarget-io	Provides a board-independent API to retarget text input/ouput to a serial UART on a kit	API Reference	✓	✓	✓
rgb-led	Provides a board-independent API to use the RGB LED on a kit	API Reference	✓	✓	✓
serial-flash	Provides a board-independent API to use the serial flash on a kit	API Reference	✓	✓	✓
psoc6pdl	Peripheral driver library for PSoC 6 devices. The library is device-independent, so can be precompiled and used for any PSoC 6 MCU device or project. Included automatically by any BSP targeting a PSoC 6 device	API Reference	✓	✓	✓
wifi-host-driver	Driver library for Cypress WLAN devices (CYW43xxx) that can be easily ported to popular RTOSs such as Amazon FreeRTOS and Mbed OS. The wifi-host-driver is included automatically by board support packages that require this driver (those with CYW43xx devices)	API Reference	✓	✓	✓
psoc6cm0p	Prebuilt application images for the Cortex M0+ CPU of the dual-CPU PSoC 6 devices. The images are provided as C arrays ready to be compiled as part of the Cortex M4 application. The Cortex M0+ application code is placed to internal flash by the Cortex M4 linker script.	See the readme file in the repository	✓	✓	✓
psoc6make	This repository provides the build recipe makefiles and scripts for building and programming PSoC 6 applications. You can build an application either through a command-line interface (CLI), the ModusToolbox IDE, or a third-party IDE.	See the readme file in the repository	✓	✓	✓
Mbed OS HAL	Mbed's hardware abstraction layer. Support for Cypress targets is available from the Mbed OS archive. This HAL is part of the Mbed ecosystem and not provided directly by Cypress.	Mbed API documentation		✓	

CYPRESS
EMBEDDED IN TOMORROW™

Peripheral Driver Library (PDL)

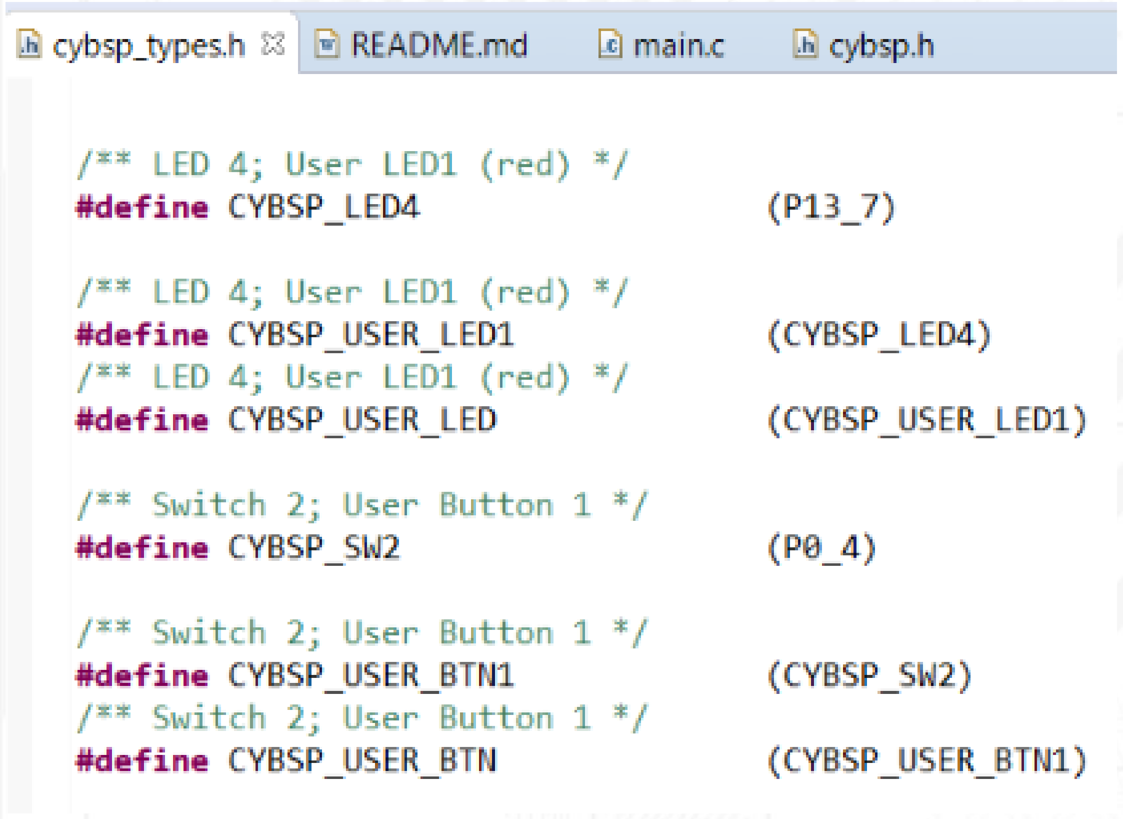
- Low level library for interacting with hardware
- Simplifies software development
- Extensive set of peripherals available

Hardware Abstraction Layer (HAL)

- Simplifies access to the low level drivers
- One call will work for drivers other than the PDL
- Does not do everything the PDL can do
- Takes care of resource allocation

Board Support Package (BSP)

- Common language for commonly used peripherals
- BSP = a board/kit
  - You can make your own and be discoverable
- Includes any low-level resources
  (e.g. the PDL library for a PSoC 6 kit)
- Has common macros for board peripherals
  (e.g. the RGB LED)
- Typically includes the design.modus file
  (sets up a default configuration for the kit)
- Uses the HAL to configure the board



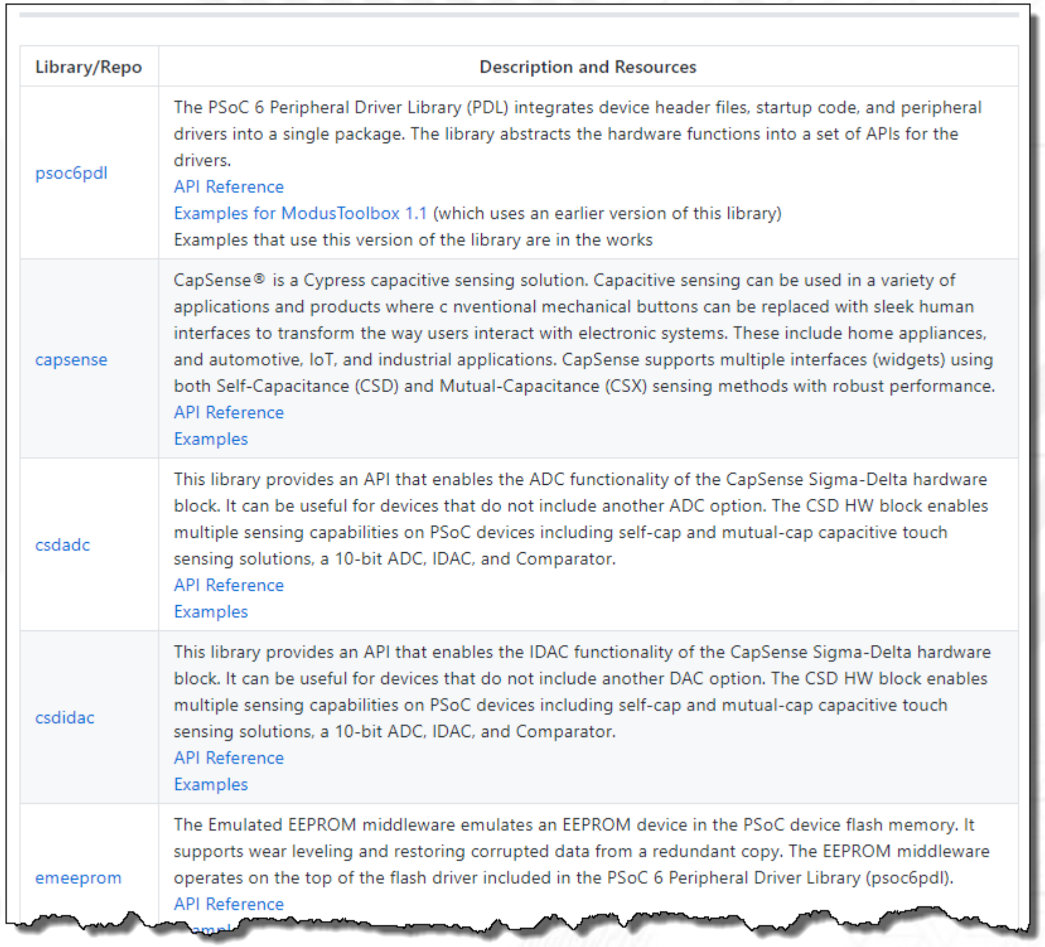
```
/** LED 4; User LED1 (red) */
#define CYBSP_LED4                     (P13_7)

/** LED 4; User LED1 (red) */
#define CYBSP_USER_LED1                (CYBSP_LED4)
/** LED 4; User LED1 (red) */
#define CYBSP_USER_LED                 (CYBSP_USER_LED1)

/** Switch 2; User Button 1 */
#define CYBSP_SW2                      (P0_4)

/** Switch 2; User Button 1 */
#define CYBSP_USER_BTN1                (CYBSP_SW2)
/** Switch 2; User Button 1 */
#define CYBSP_USER_BTN                 (CYBSP_USER_BTN1)
```

HAL, BSP and PDL, what do you use?

- BSPs use the HAL to configure boards
- HAL is intended to be the future
  - Makes the application device agnostic.
- PDL and HAL are 100% compatible – you can mix and match
  - But it's possible to use both to configure the same thing (developer's responsibility)
- Intent is that configurator will flag which peripherals it uses
  - So HAL won't step on configuration code

Middleware

- Each Cypress library is available independently on GitHub
  - Some ecosystem libraries just live in that ecosystem
- Connectivity middleware
  - BT SDK (WICED), AWS IoT, Wi-Fi, BLE
- PSoC 6 middleware
  - CapSense, USB Dev, Device firmware upgrade, etc.
- Bottom line, there is a LOT
  - Some work for multiple ecosystems
  - Some are specific to a particular ecosystem
- GitHub landing page for [PSoC 6 Middleware]



Library/Repo	Description and Resources
psoc6pdl	The PSoC 6 Peripheral Driver Library (PDL) integrates device header files, startup code, and peripheral drivers into a single package. The library abstracts the hardware functions into a set of APIs for the drivers. API Reference Examples for ModusToolbox 1.1 (which uses an earlier version of this library) Examples that use this version of the library are in the works
capsense	CapSense® is a Cypress capacitive sensing solution. Capacitive sensing can be used in a variety of applications and products where c nventional mechanical buttons can be replaced with sleek human interfaces to transform the way users interact with electronic systems. These include home appliances, and automotive, IoT, and industrial applications. CapSense supports multiple interfaces (widgets) using both Self-Capacitance (CSD) and Mutual-Capacitance (CSX) sensing methods with robust performance. API Reference Examples
csdadc	This library provides an API that enables the ADC functionality of the CapSense Sigma-Delta hardware block. It can be useful for devices that do not include another ADC option. The CSD HW block enables multiple sensing capabilities on PSoC devices including self-cap and mutual-cap capacitive touch sensing solutions, a 10-bit ADC, IDAC, and Comparator. API Reference Examples
csdidac	This library provides an API that enables the IDAC functionality of the CapSense Sigma-Delta hardware block. It can be useful for devices that do not include another DAC option. The CSD HW block enables multiple sensing capabilities on PSoC devices including self-cap and mutual-cap capacitive touch sensing solutions, a 10-bit ADC, IDAC, and Comparator. API Reference Examples
emeeprom	The Emulated EEPROM middleware emulates an EEPROM device in the PSoC device flash memory. It supports wear leveling and restoring corrupted data from a redundant copy. The EEPROM middleware operates on the top of the flash driver included in the PSoC 6 Peripheral Driver Library (psoc6pdl). API Reference

CYPRESS
EMBEDDED IN TOMORROW™

Code Examples

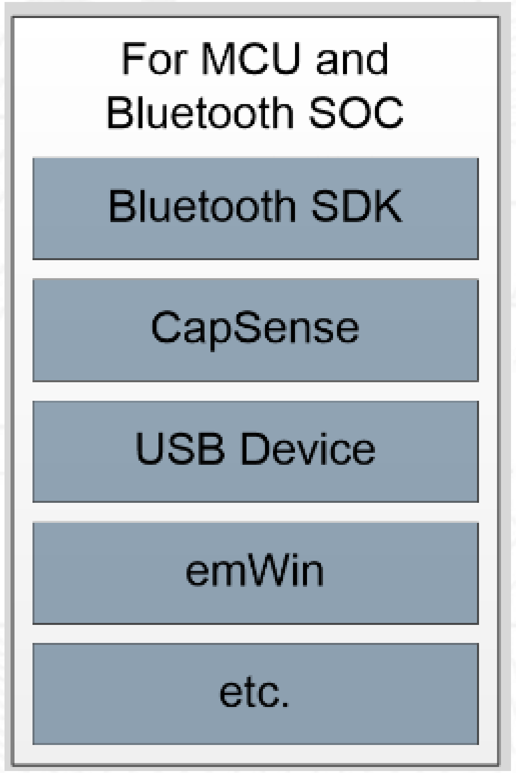
Item	Details
All ModusToolbox Examples	A GitHub landing page with information about the examples and links to various collections.
Bluetooth SDK Examples	A single repository that contains all the BT SDK examples.
PSoC 6 MCU Examples	This search link displays a list of all MCU related examples.
AWS IoT Greengrass Examples	These repositories contain all the AwS IoT-related examples in the Mbed OS ecosystem.
AWS IoT Client Examples	
Mbed OS Examples	This search link displays a list of all Mbed-OS examples on the Cypress GitHub site.

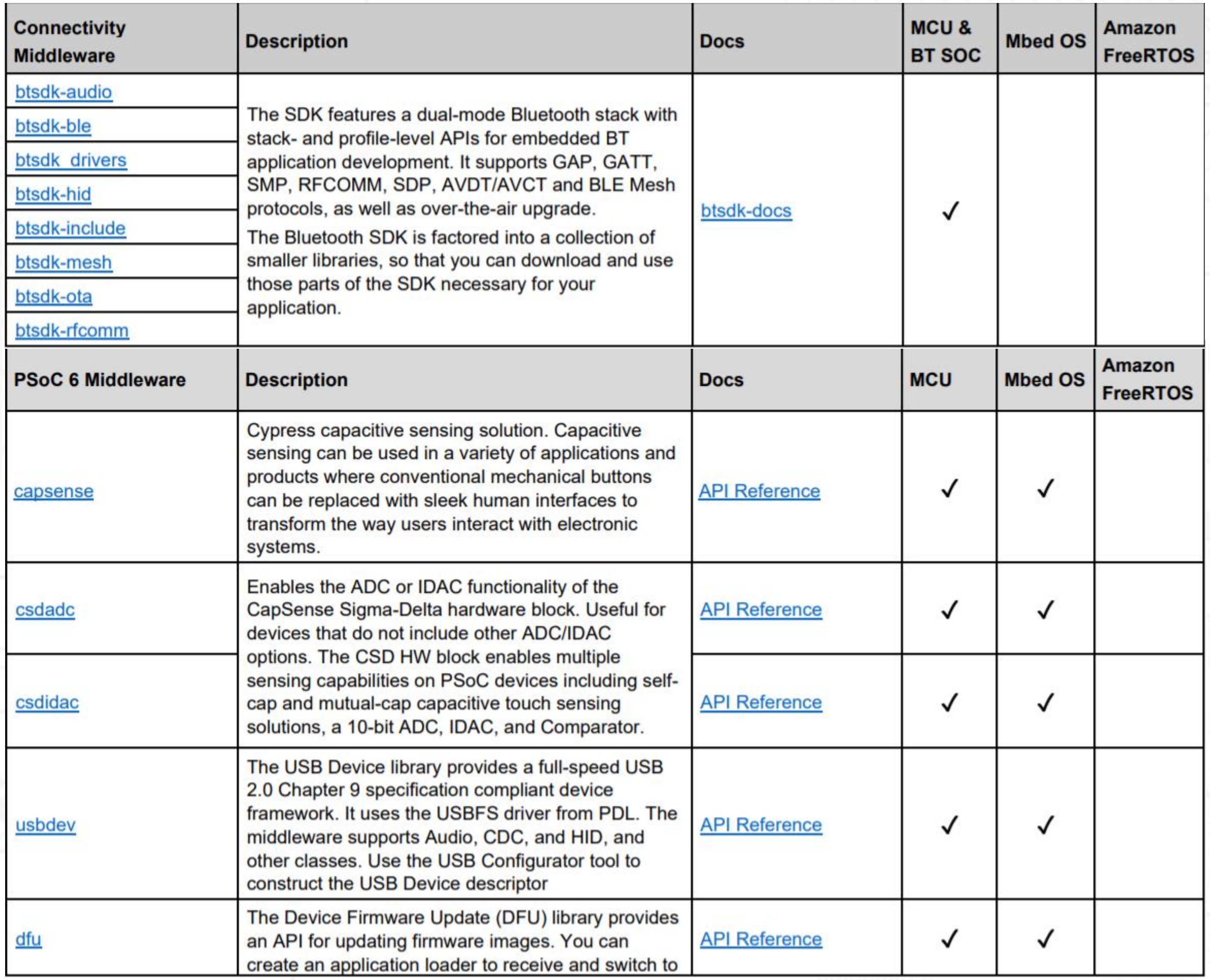
- All on GitHub!
- In general, each example will live in its own repository
  - exception for an example with multiple projects

Targeted Ecosystems

PSoC 6 MCU and Bluetooth SOC Ecosystem

- All low-level resources
- Most if not all BSPs



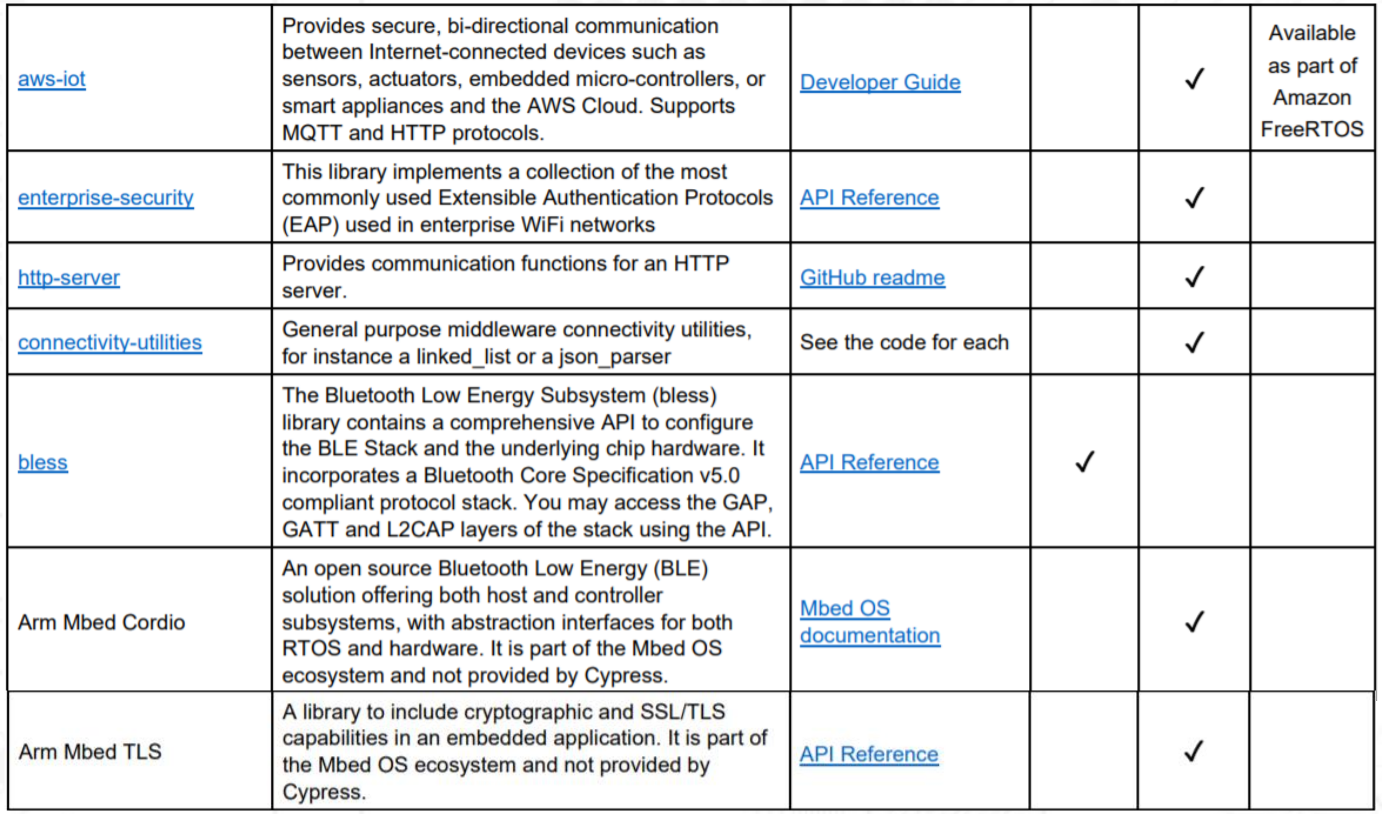


Connectivity Middleware	Description	Docs	MCU & BT SOC	Mbed OS	Amazon FreeRTOS
btsdk-audio	The SDK features a dual-mode Bluetooth stack with stack- and profile-level APIs for embedded BT application development. It supports GAP, GATT, SMP, RFCOMM, SDP, AVDT/AVCT and BLE Mesh protocols, as well as over-the-air upgrade. The Bluetooth SDK is factored into a collection of smaller libraries, so that you can download and use those parts of the SDK necessary for your application.	btsdk-docs	✓		
btsdk-ble					
btsdk_drivers					
btsdk-hid					
btsdk-include					
btsdk-mesh					
btsdk-ota					
btsdk-rfcomm					

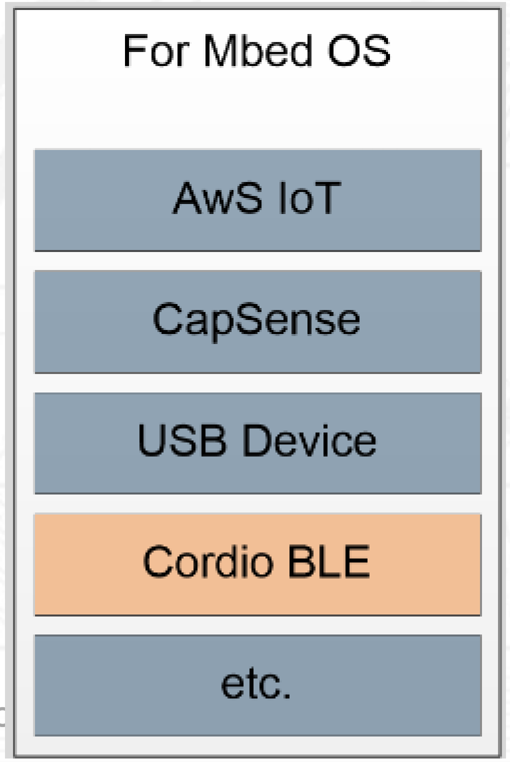
PSoC 6 Middleware	Description	Docs	MCU	Mbed OS	Amazon FreeRTOS
capsense	Cypress capacitive sensing solution. Capacitive sensing can be used in a variety of applications and products where conventional mechanical buttons can be replaced with sleek human interfaces to transform the way users interact with electronic systems.	API Reference	✓	✓	
csdadc	Enables the ADC or IDAC functionality of the CapSense Sigma-Delta hardware block. Useful for devices that do not include other ADC/IDAC options. The CSD HW block enables multiple sensing capabilities on PSoC devices including self-cap and mutual-cap capacitive touch sensing solutions, a 10-bit ADC, IDAC, and Comparator.	API Reference	✓	✓	
csdidac		API Reference	✓	✓	
usbdev	The USB Device library provides a full-speed USB 2.0 Chapter 9 specification compliant device framework. It uses the USBFS driver from PDL. The middleware supports Audio, CDC, and HID, and other classes. Use the USB Configurator tool to construct the USB Device descriptor	API Reference	✓	✓	
dfu	The Device Firmware Update (DFU) library provides an API for updating firmware images. You can create an application loader to receive and switch to	API Reference	✓	✓	

Mbed OS Ecosystem

- All low-level resources
- BSPs for Mbed supported boards
- All PSoC 6 middleware
  - CapSense, etc.
- Support for AWS IoT inside Mbed

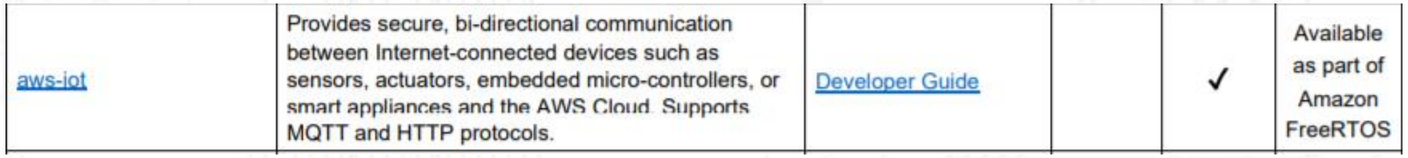


aws-iot	Provides secure, bi-directional communication between Internet-connected devices such as sensors, actuators, embedded micro-controllers, or smart appliances and the AWS Cloud. Supports MQTT and HTTP protocols.	Developer Guide		✓	Available as part of Amazon FreeRTOS
enterprise-security	This library implements a collection of the most commonly used Extensible Authentication Protocols (EAP) used in enterprise WiFi networks	API Reference		✓	
http-server	Provides communication functions for an HTTP server.	GitHub readme		✓	
connectivity-utilities	General purpose middleware connectivity utilities, for instance a linked_list or a json_parser	See the code for each		✓	
bless	The Bluetooth Low Energy Subsystem (bless) library contains a comprehensive API to configure the BLE Stack and the underlying chip hardware. It incorporates a Bluetooth Core Specification v5.0 compliant protocol stack. You may access the GAP, GATT and L2CAP layers of the stack using the API.	API Reference	✓		
Arm Mbed Cordio	An open source Bluetooth Low Energy (BLE) solution offering both host and controller subsystems, with abstraction interfaces for both RTOS and hardware. It is part of the Mbed OS ecosystem and not provided by Cypress.	Mbed OS documentation		✓	
Arm Mbed TLS	A library to include cryptographic and SSL/TLS capabilities in an embedded application. It is part of the Mbed OS ecosystem and not provided by Cypress.	API Reference		✓	



Amazon FreeRTOS Ecosystem

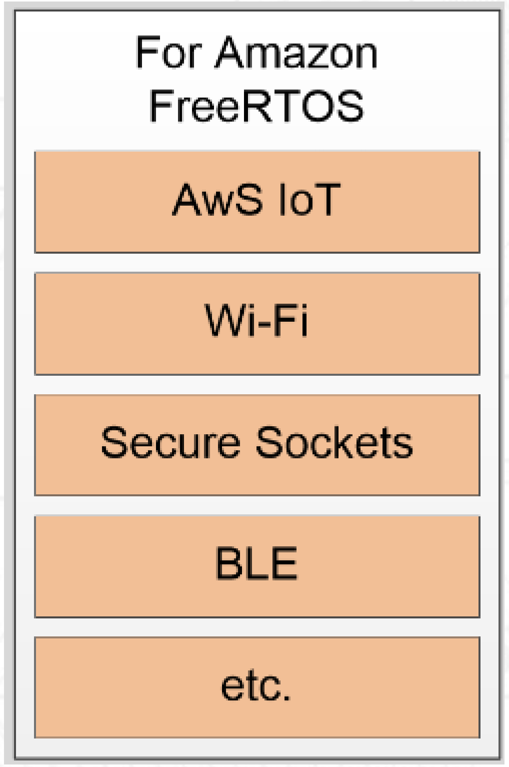
- All low level resources
- Two BSPs Qualified
  - CY8CPROTO-062-4343W
  - CY8CKIT-062-WIFI-BT
- No PSoC 6 MCU middleware
  - E.g.CapSense
- Rich set of middleware
  - All from 3rd parties, not Cypress
  - [Amazon FreeRTOS Libraries]

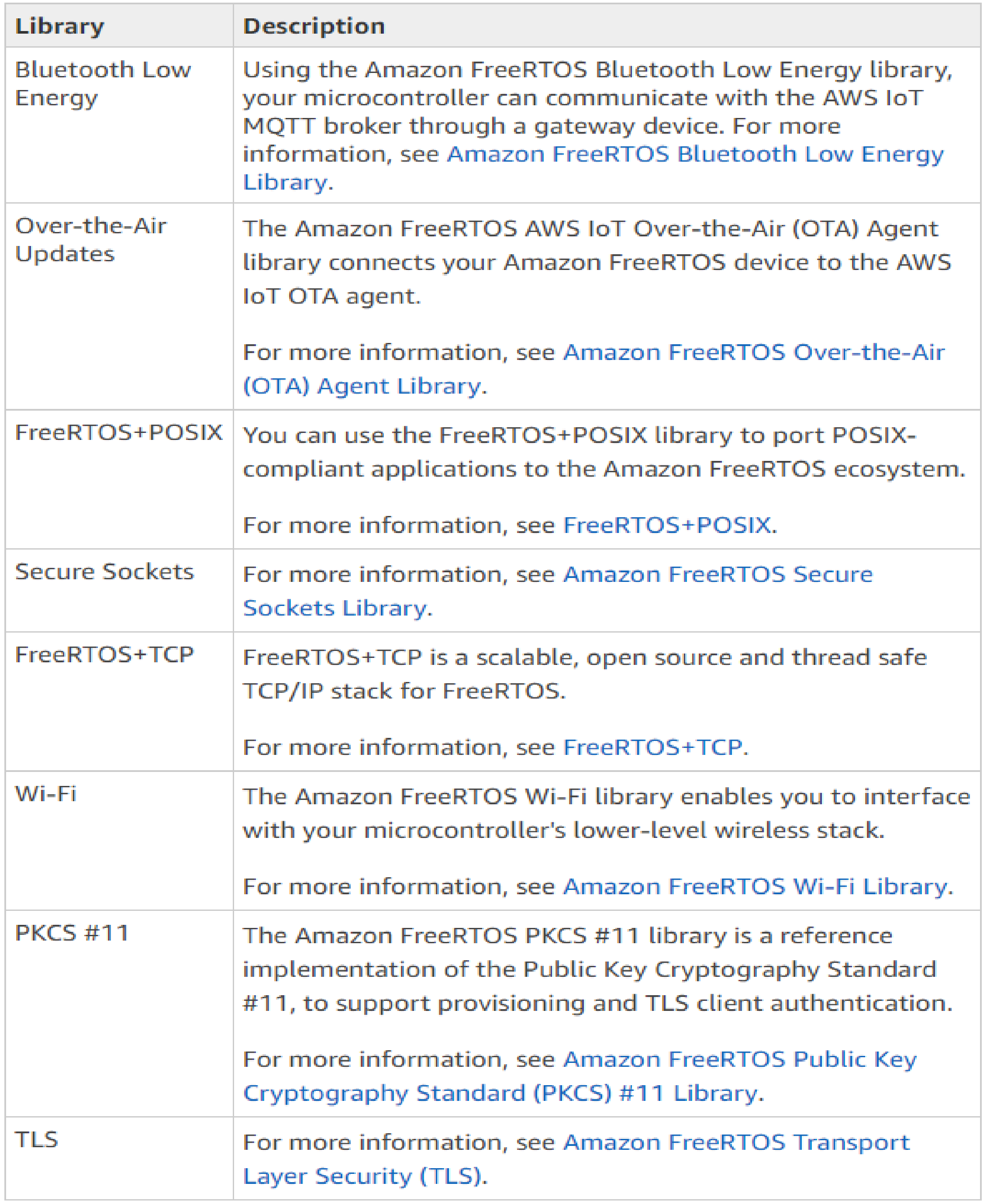


aws-iot	Provides secure, bi-directional communication between Internet-connected devices such as sensors, actuators, embedded micro-controllers, or smart appliances and the AWS Cloud. Supports MQTT and HTTP protocols.	Developer Guide		✓	Available as part of Amazon FreeRTOS

Amazon FreeRTOS libraries

- It is a rich ecosystem





Library	Description
Bluetooth Low Energy	Using the Amazon FreeRTOS Bluetooth Low Energy library, your microcontroller can communicate with the AWS IoT MQTT broker through a gateway device. For more information, see Amazon FreeRTOS Bluetooth Low Energy Library.
Over-the-Air Updates	The Amazon FreeRTOS AWS IoT Over-the-Air (OTA) Agent library connects your Amazon FreeRTOS device to the AWS IoT OTA agent. For more information, see Amazon FreeRTOS Over-the-Air (OTA) Agent Library.
FreeRTOS+POSIX	You can use the FreeRTOS+POSIX library to port POSIX-compliant applications to the Amazon FreeRTOS ecosystem. For more information, see FreeRTOS+POSIX.
Secure Sockets	For more information, see Amazon FreeRTOS Secure Sockets Library.
FreeRTOS+TCP	FreeRTOS+TCP is a scalable, open source and thread safe TCP/IP stack for FreeRTOS. For more information, see FreeRTOS+TCP.
Wi-Fi	The Amazon FreeRTOS Wi-Fi library enables you to interface with your microcontroller's lower-level wireless stack. For more information, see Amazon FreeRTOS Wi-Fi Library.
PKCS #11	The Amazon FreeRTOS PKCS #11 library is a reference implementation of the Public Key Cryptography Standard #11, to support provisioning and TLS client authentication. For more information, see Amazon FreeRTOS Public Key Cryptography Standard (PKCS) #11 Library.
TLS	For more information, see Amazon FreeRTOS Transport Layer Security (TLS).

Resources

- ModusToolbox User Guide
- ModusToolbox Software Overview
- Running ModusToolbox from the Command Line
- Cypress Github Landing Page
- Mbed OS Cypress Page
- Amazon FreeRTOS Cypress Repo
- Managing the Makefile in ModusToolbox

CYPRESS
EMBEDDED IN TOMORROW™

CYPRESS®
EMBEDDED IN TOMORROW™