



ModusToolbox™ IDE

User Guide

Document Number: 002-24375 Rev *B

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
<http://www.cypress.com>

© Cypress Semiconductor Corporation, 2018-2019. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, ModusToolbox, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.

Contents



1. Introduction.....	5
Overview	5
References	6
Document Conventions	6
Revision History	6
2. Getting Started.....	7
Launch ModusToolbox IDE.....	7
Open New Application Dialog	8
Create PSoC 6 MCU Starter Application	9
Create WICED Bluetooth Application	11
Build Starter Application	15
Program Starter Application.....	15
Download/Import Code Examples	16
3. Mbed OS to ModusToolbox Flow	18
Install and Configure Mbed CLI	18
Switch Kit to DAPLink Mode.....	20
Create Application in Mbed CLI	20
Compile Application in Mbed CLI (Optional).....	21
Export Application from Mbed CLI	21
Import Application into ModusToolbox IDE	21
Configure ModusToolbox IDE	23
Build, Program, Debug Application	25
4. IDE Description.....	27
Overview	27
Project Explorer.....	28
Quick Panel	29
5. Installation Description.....	30
Directory Structure	30
Documentation	30
Tools.....	31
6. Configure Applications	33
Modify Code	33
Use Configurators	34
Use Tools.....	38
Rename Application	39

- 7. Build Applications 40**
 - Build with Make 40
 - Build with IDE 40
 - GCC Version 42

- 8. Program and Debug 43**
 - PSoC 6 Programming/Debugging 44
 - Bluetooth Programming/Debugging 48
 - Mbed OS Programming/Debugging 49

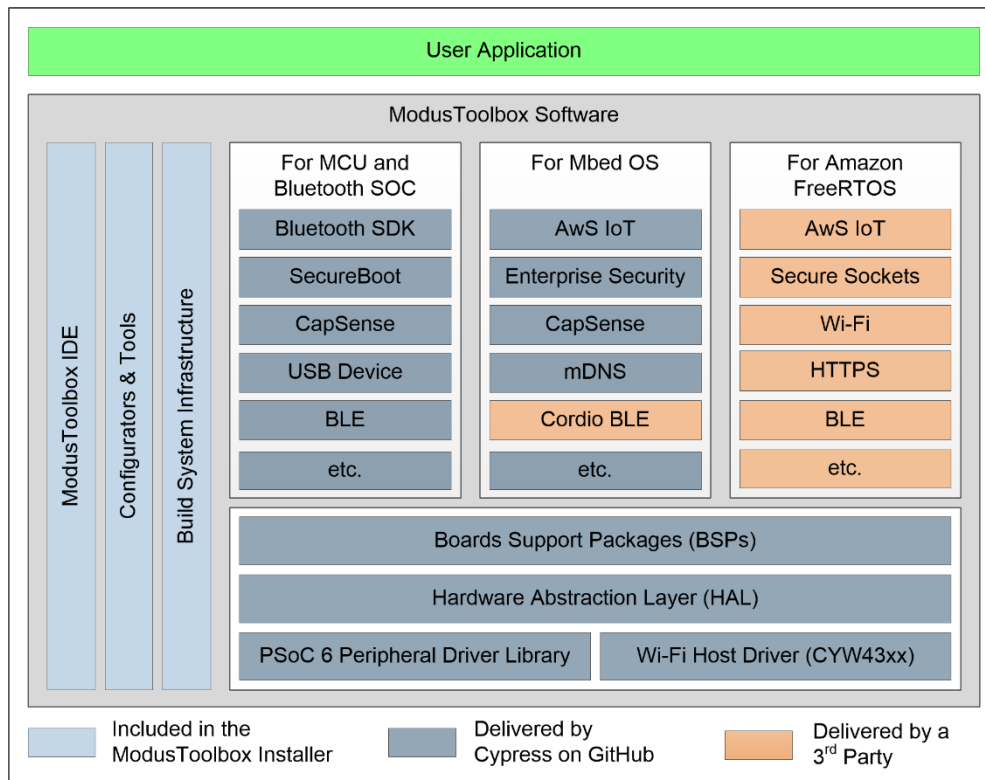
1. Introduction



Overview

ModusToolbox™ software is a set of tools that enable you to integrate Cypress devices into your existing development methodology. One of the tools is a multi-platform, Eclipse-based Integrated Development Environment (IDE) called the ModusToolbox IDE that supports application configuration and development.

The following shows a high-level view of the tools included in the ModusToolbox software. For a more comprehensive description of the ModusToolbox software, refer to the *ModusToolbox Software Overview* document.



As shown, the ModusToolbox ecosystem includes IoT development platforms like Mbed™ OS and Amazon FreeRTOS™, in addition to the software installed on your computer.

This document provides information about creating applications as well as building, programming, and debugging them. This guide primarily covers the ModusToolbox IDE aspects of these concepts. It also covers various aspects of the software installed along with the IDE, where applicable. For information about using Mbed OS, refer to the [Mbed OS to ModusToolbox Flow](#) chapter in this document. For information about using Amazon FreeRTOS, refer to the [their website](#).

References

The following documents should be referenced as needed for more information about a particular topic, For more information, see [Documentation](#).

- [ModusToolbox Installation Guide](#)
- [ModusToolbox Software Overview](#)
- Configurator Documentation (open from a particular Configurator)
- Eclipse Documentation (available from Eclipse)
- [Eclipse Survival Guide](#)
- [KitProg3 User Guide](#)

Document Conventions

The following table lists the conventions used throughout this guide:

Convention	Usage
Courier New	Displays file locations and source code: <code>C:\...cd\icc\, user entered text</code>
<i>Italics</i>	Displays file names and reference documentation: <i>sourcefile.hex</i>
[bracketed, bold]	Displays keyboard commands in procedures: [Enter] or [Ctrl] [C]
File > New Application	Represents menu paths: File > New Application > Clone
Bold	Displays commands, menu paths and selections, and icon names in procedures: Click the Debugger icon, and then click Next .
Text in gray boxes	Displays cautions or functionality unique to ModusToolbox software.

Revision History

Document Title: ModusToolbox™ IDE User Guide		
Document Number: 002-24375		
Revision	Date	Description of Change
**	11/21/18	New document.
*A	2/22/19	Updates for ModusToolbox version 1.1.
*B	10/15/2019	Updates for ModusToolbox version 2.0.

2. Getting Started



This section provides a basic walkthrough for how to create a couple starter applications using the IDE, selecting either a kit or custom hardware. It also covers how to build and program them using the IDE and basic launch configurations supplied for the applications.

- [Launch ModusToolbox IDE](#)
- [Open New Application Dialog](#)
- [Create PSoC 6 MCU Starter Application](#)
- [Create WICED Bluetooth Application](#)
- [Build Starter Application](#)
- [Program Starter Application](#)
- [Download/Import Code Examples](#)

Launch ModusToolbox IDE

The ModusToolbox IDE is installed in the following directory in Windows, by default:

```
<user_home>\ModusToolbox\ide_2.0\eclipse\
```

For other operating systems, the installation directory will vary, based on where the software was installed.

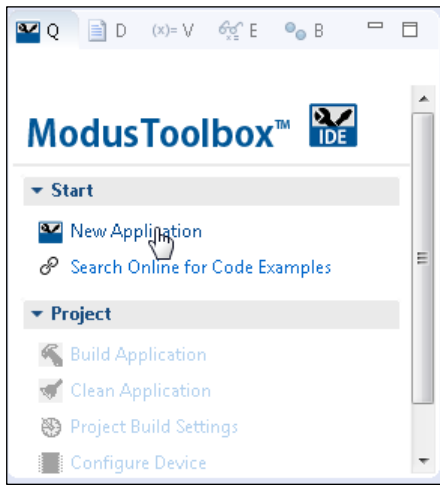
Note If the software is not installed in the default location, you will need to set an environment variable. Refer to the [ModusToolbox Installation Guide](#) for details.

Run the "ModusToolbox" executable file to launch the IDE as applicable for your operating system. Refer to the [Modus Toolbox Installation Guide](#) for more information.

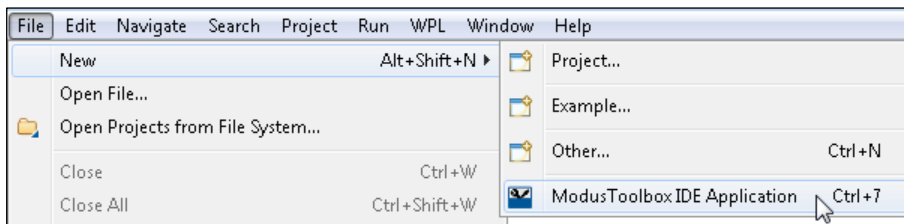
When launching the ModusToolbox IDE, it provides an option to select the workspace location on your machine. This location is used by the IDE for creating and storing the files as part of application creation for a particular platform. The default workspace location is a folder called "mtw" in your home directory. You may add additional folders under the "mtw" folder or to choose any other location for each workspace.

Open New Application Dialog

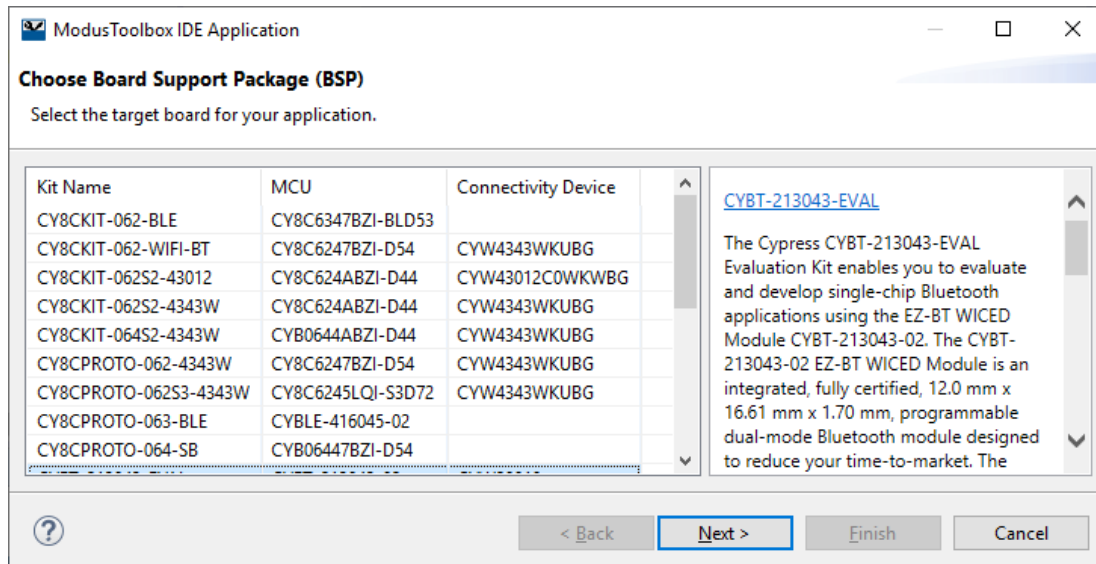
Click the **New Application** link in the ModusToolbox IDE Quick Panel.



You can also select **File > New > ModusToolbox IDE Application**.



The New Application dialog provides several starter applications for use with different development kits. Refer to the *ModusToolbox IDE Release Notes* for the supported platforms.

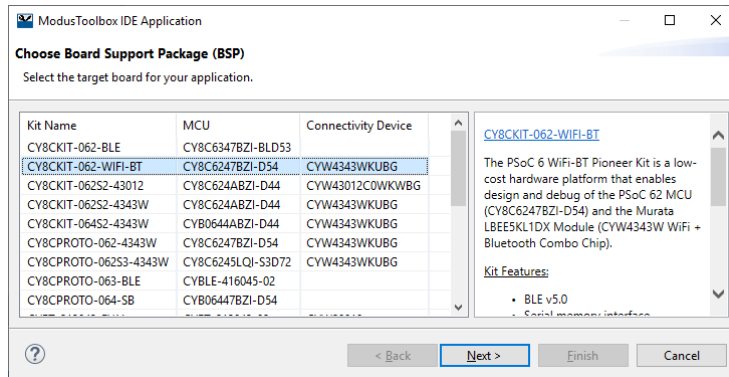


Create PSoC 6 MCU Starter Application

This section provides a walkthrough for creating a PSoC 6 MCU application.

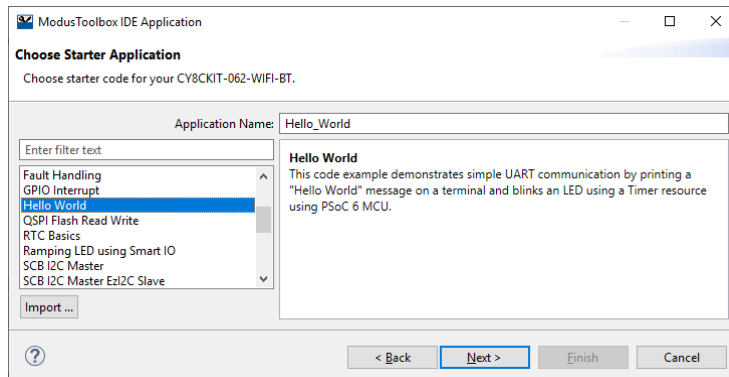
Choose Board Support Package

The ModusToolbox IDE Application dialog displays a list of boards, showing the Kit Name, MCU, and Connectivity Device (if applicable). As you select each of the kits shown, the description for that kit displays on the right. For this example, select the CY8CKIT-062-WIFI-BT kit.



Choose Starter Application

Click **Next >** to open the Starter Application page. This page lists various starter applications available for the selected kit. As you select a starter application, a description displays on the right.



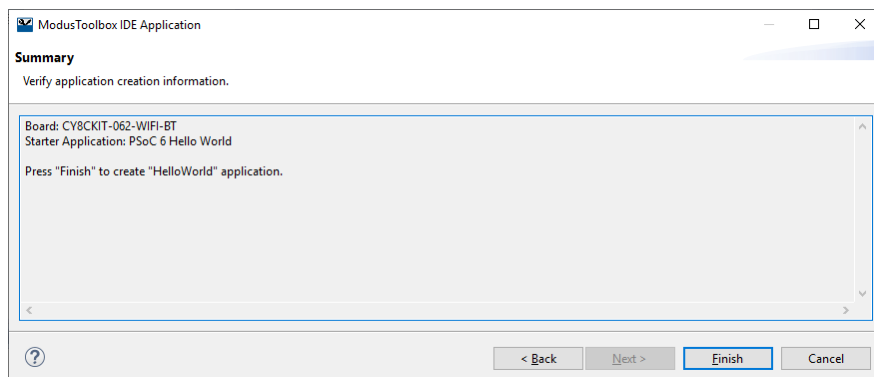
For this example:

- Select the "Hello World" application from the list.
- Type a name for your application. Do not use spaces in the application name. In this case, use "HelloWorld" as the **Application name**.

Note You can use the **Import...** button to select other examples you downloaded from the web or received from a colleague. In the Open dialog, select only examples that are supported by the BSP you selected for this application. Then, the example will be shown in the New Application dialog with all the other Starter Applications. See also [Import Code Examples](#) for additional details.

Review Summary

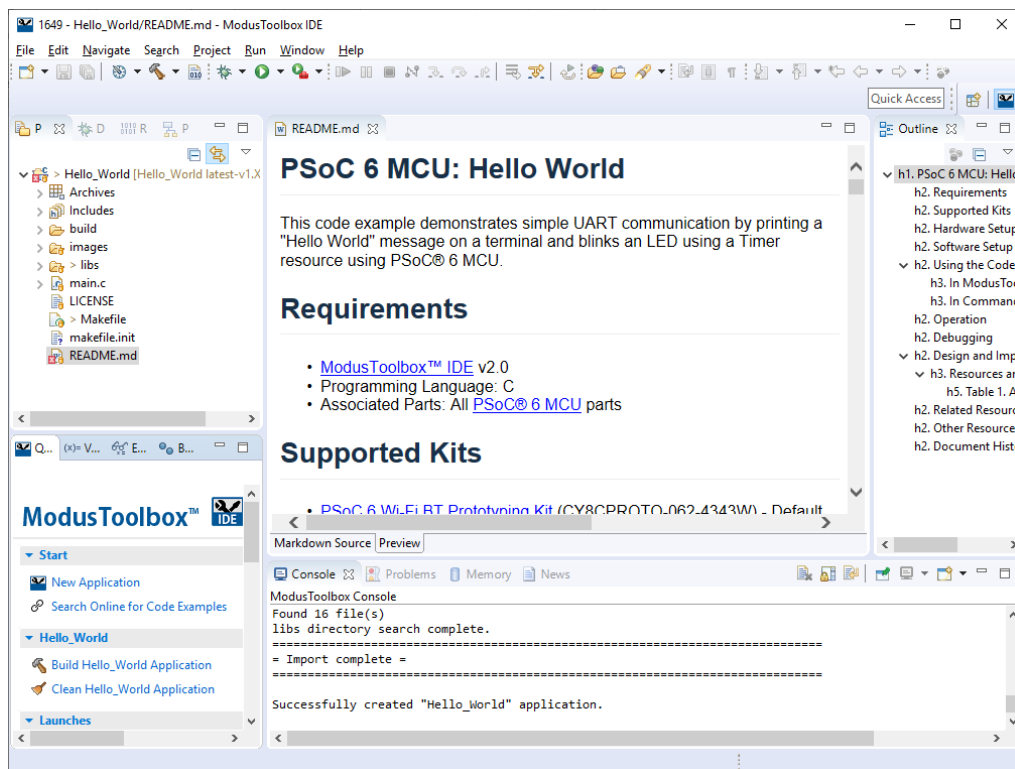
Click **Next >** to open the Summary page. This page shows the various options chosen for this application. Review them to ensure they are correct.



Click **Finish** to create the application. After a few moments, the ModusToolbox IDE displays progress information.

Note The application creation process performs a `git clone` operation, and downloads the selected application from the Cypress GitHub website. Depending on the selected application, this process can take several minutes.

When complete, the application's project folders display in the [Project Explorer](#).



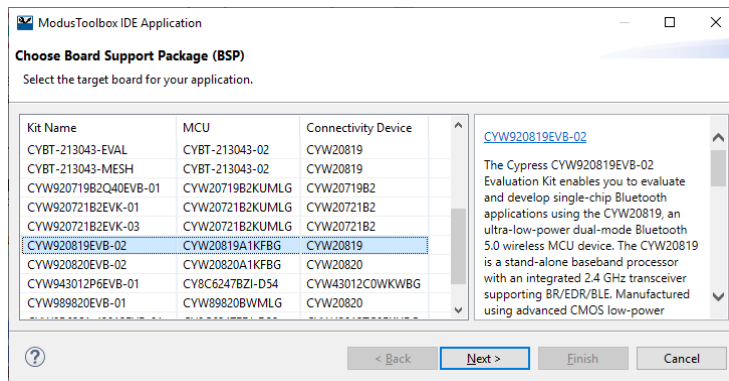
Create WICED Bluetooth Application

This section provides a walkthrough for creating a WICED Bluetooth application.

Note For Bluetooth applications, you must first create a separate project called `wiced_btSDK`, which contains the SDK, BSPs, and libraries that are shared among all Bluetooth applications in a workspace. This application only needs to be created once per workspace.

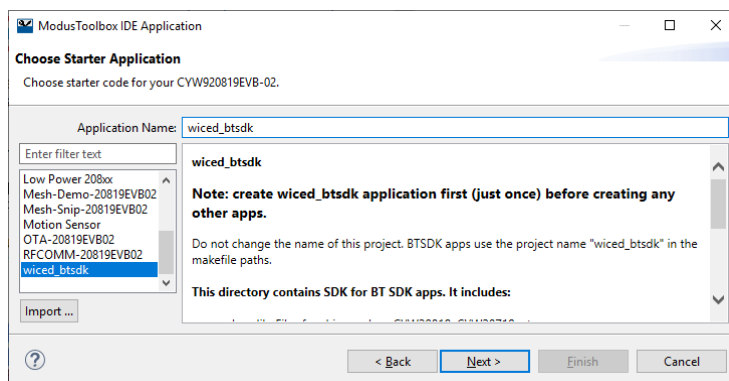
Choose Board Support Package

The ModusToolbox IDE Application dialog displays a list of boards, showing the Kit Name, MCU, and Connectivity Device (if applicable). As you select each of the kits shown, the description for that kit displays on the right. For this example, select the `CYW920819EVB-02` Evaluation kit. For the `wiced_btSDK` project, you can choose any board.



Choose wiced_btSDK Starter Application

Click **Next >** to open the Starter Application page. This page lists various starter applications available for the selected kit. As you select a starter application, a description displays on the right.

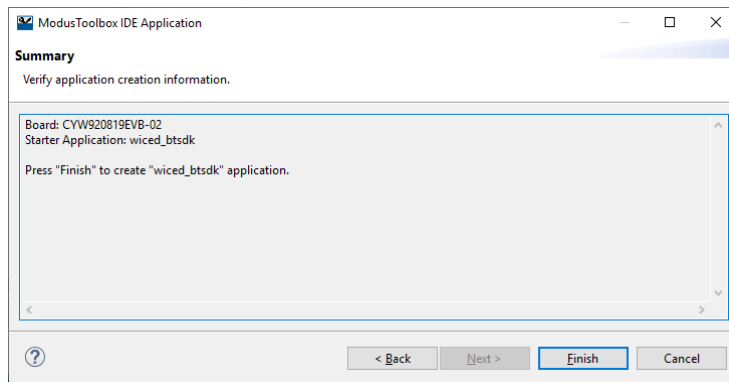


For this example:

- Select the "wiced_btSDK" application from the list.
- Do not change the name of this project. Bluetooth applications use the project name "wiced_btSDK" in the makefile paths.

Review Summary

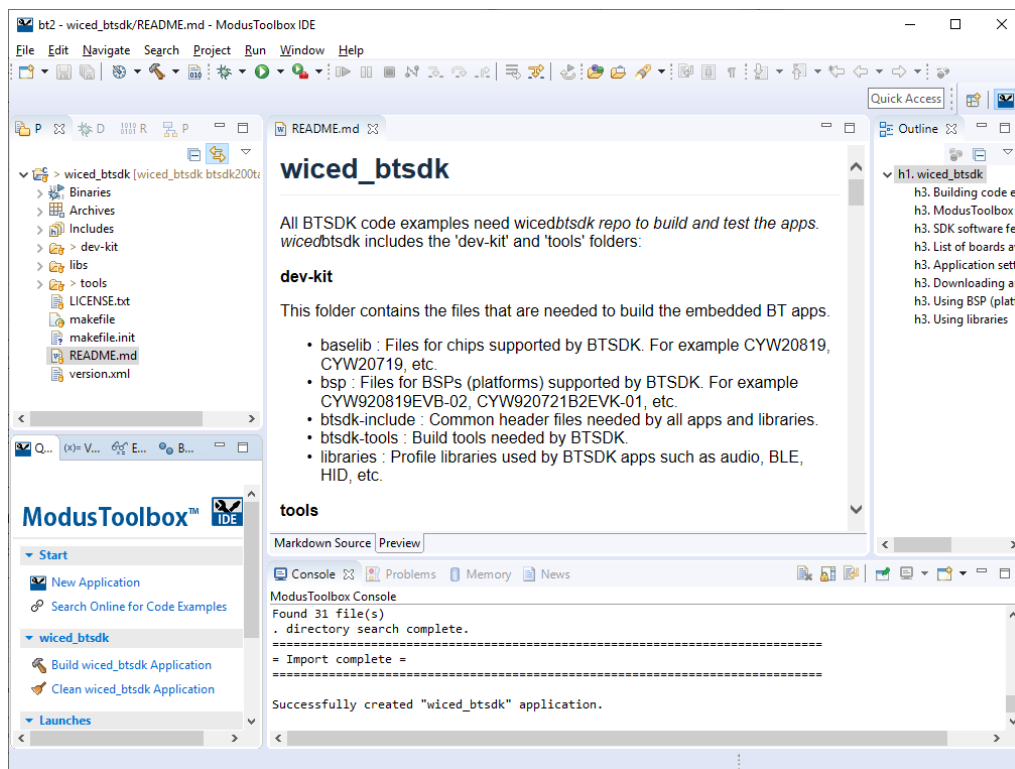
Click **Next >** to open the Summary page. This page shows the various options chosen for this application. Review them to ensure they are correct.



Click **Finish** to create the application. After a few moments, the ModusToolbox IDE displays progress information.

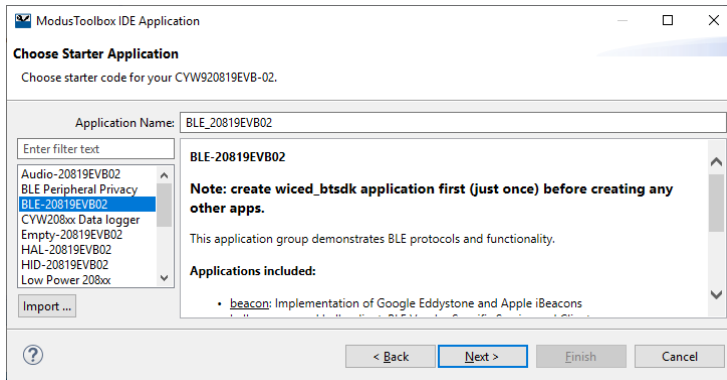
Note The application creation process performs a `git clone` operation, and downloads the selected application from the Cypress GitHub website. Depending on the selected application, this process can take several minutes.

When complete, the application's project folders display in the [Project Explorer](#).



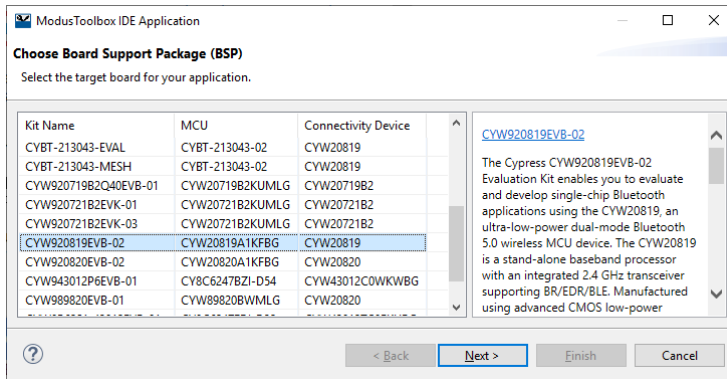
Choose a Bluetooth Starter Application

Repeat the new application process to create a BLE-20819EVB02 application in the same workspace as the wiced_btsdk project.



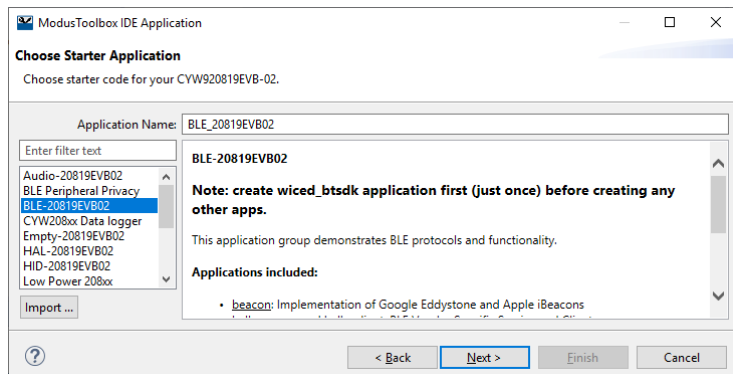
Choose Board Support Package

For this example, select the CYW920819EVB-02 Evaluation kit.



Choose Starter Application

Click **Next >** to open the Starter Application page.

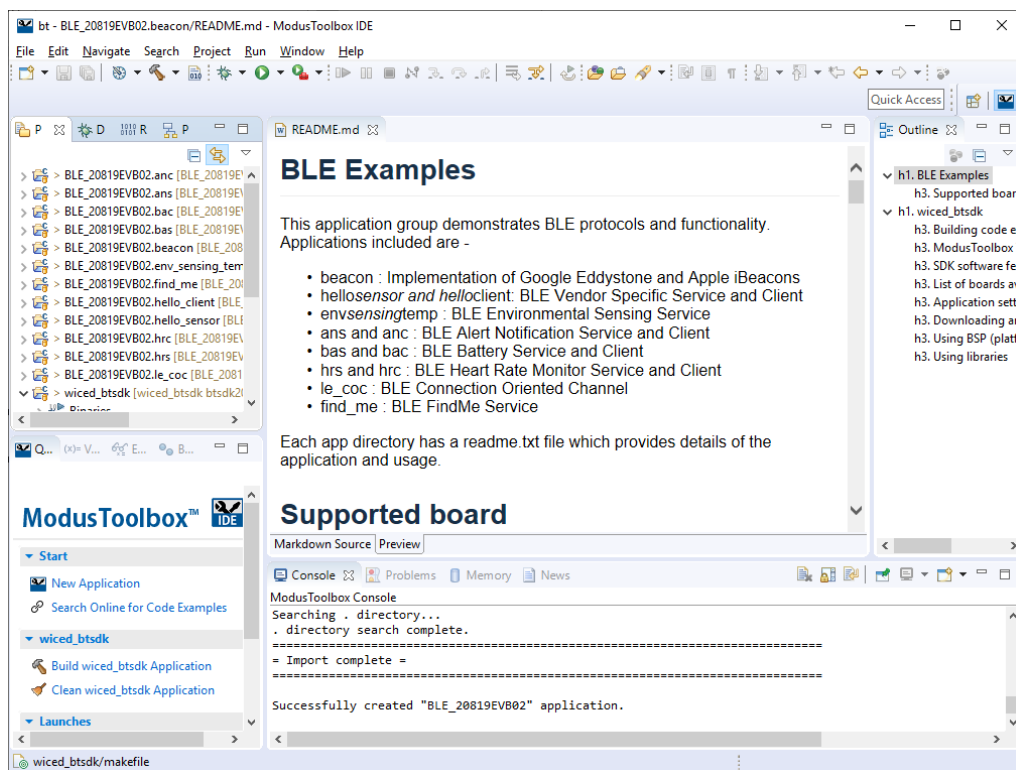


For this example:

- Select the "BLE-20819EVB02" application from the list.
- Type a name for your application. Do not use spaces in the application name. In this case, use "BLE_20819EVB02" as the **Application name**.

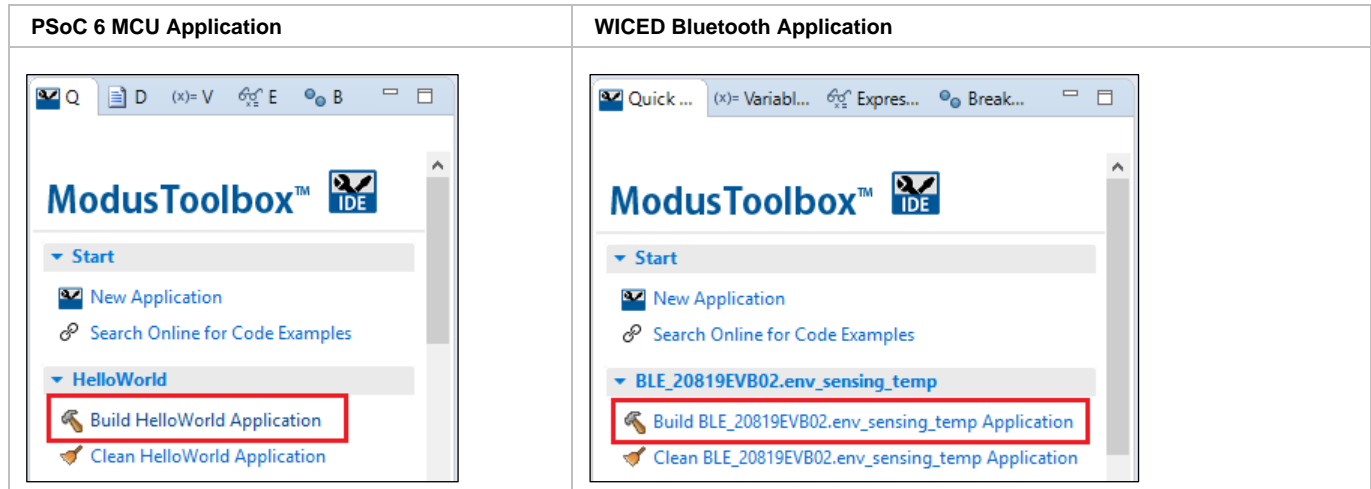
When complete, the application's project folders display in the [Project Explorer](#).

Note Many Bluetooth starter applications contain multiple projects. For example, the BLE-20819EVB02 starter application contains projects for BLE services such as anc, ans, bac, bas, beacon, etc.



Build Starter Application

After loading an application, build it to generate the necessary files. Select a project. Then, in the **Quick Panel**, click the "<project> Application" link.



Messages display in the Console, indicating whether the build was successful or not. For more details about building applications and the various Consoles available, see the [Build Applications](#) chapter.

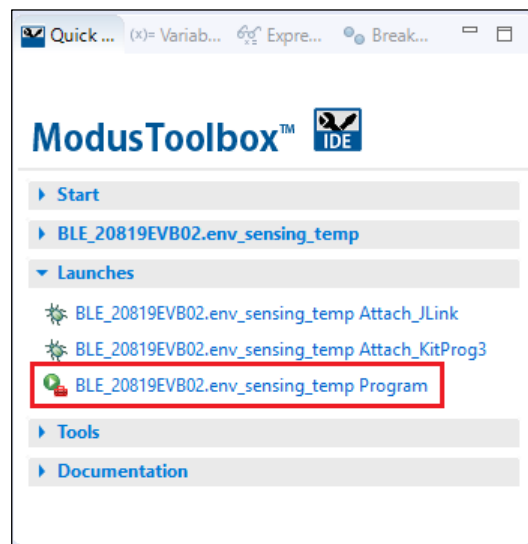
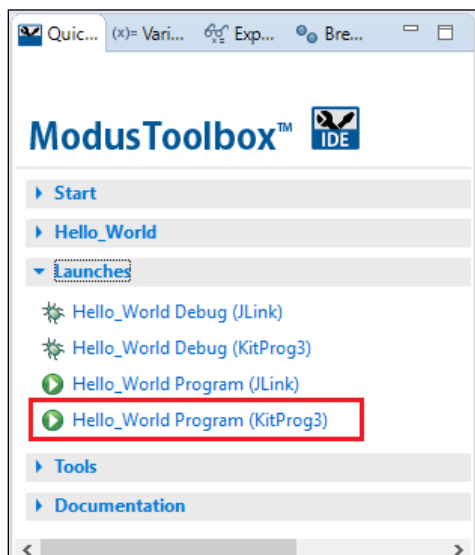
Program Starter Application

There are many more details about programming an application. This section only covers it briefly. For more detailed information, see the [Program and Debug](#) chapter.

In the Project Explorer, select the desired project. Then, in the **Quick Panel**, click the "<app-name> Program (KitProg3)" link for a PSoC 6 MCU application, and "<app-name> Program" for a WICED Bluetooth application.

PSoC 6 MCU Application

WICED Bluetooth Application

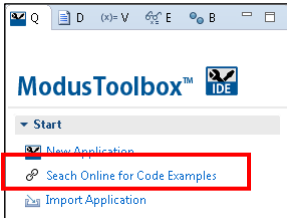


Download/Import Code Examples

Cypress provides many code examples. These examples allow you to explore the capabilities provided by the SDK, create applications based on them, examine the source code demonstrated in them, and read their associated documentation.

Download from GitHub

The **Quick Panel** provides a link to access online code examples. Click the "Search Online for Code Examples" link.



This opens a web browser to the GitHub repository to select and download the appropriate examples.

Code Examples for ModusToolbox Software

There are hundreds of code examples available - so many we can't put them all in one repository. Use the links below to find the example you want, learn more about each repo, and discover how to bring that code example into your development environment.

Code Examples for ModusToolbox Software

Repo	Description
PSoC 6 SDK Examples	A single repo with many examples that demonstrate the peripherals and functionality of the PSoC® 6 MCU. These examples support ModusToolbox 1.1
Bluetooth SDK Examples	A single repo with demo applications and snippets for various bluetooth devices and kits, including mesh network examples.
AWS IoT Examples	This search link displays Amazon Web Services examples: publisher and subscriber, and greengrass, using the Mbed OS ecosystem.
Mbed OS Examples	This search link displays a list of all Mbed-OS examples on the Cypress GitHub site, including CapSense, the emWin library, AWS IoT, and more.

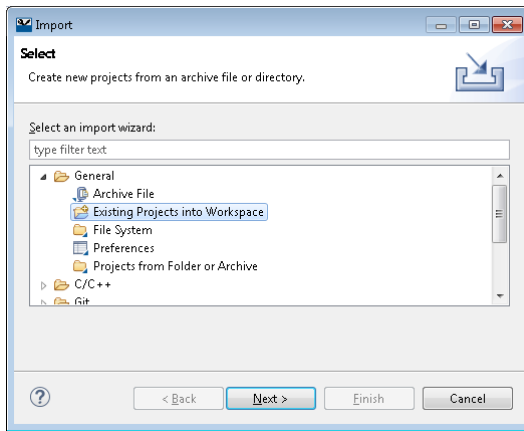
The AWS examples rely on the [AWS IoT Client Library](#). The examples will link that library automatically.

Import Code Examples

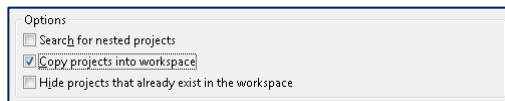
Whether you've downloaded an example, or received one from a colleague, Cypress recommends one of the following methods to import the example into the ModusToolbox IDE:

- Use the [New Application wizard](#) to create a new application, and in that process select the **Import...** button to select a folder that contains the application to import.

- If you have an Eclipse-ready code example (for example, a project exported from Eclipse) that you want to import into the ModusToolbox IDE, use the standard **File > Import** process. Note the following:
 - On the Import dialog, select **General > Existing Projects into Workspace**.



- On the next page, enable the **Copy projects into Workspace** check box.



Note There are various ways to import examples into Eclipse. If you prefer a different method, make sure that all of the project files are copied into the workspace directory.

3. Mbed OS to ModusToolbox Flow



For ModusToolbox 2.0, Cypress has enabled several kits to be used in the Mbed ecosystem. For more information, refer to the Cypress webpage on the Mbed OS website: <https://os.mbed.com/teams/Cypress/>.

These kits support various connectivity applications that you cannot create from the ModusToolbox IDE. You must use the Mbed OS flow for these types of applications. This section provides the necessary steps to import, build, program, and debug Mbed OS applications in the ModusToolbox IDE. The main steps include:

- [Install and Configure Mbed CLI](#)
- [Switch Kit to DAPLink Mode](#)
- [Create Application in Mbed CLI](#)
- [Compile Application in Mbed CL \(Optional\)](#)
- [Export Application from Mbed CLI](#)
- [Import Application into ModusToolbox IDE](#)
- [Configure ModusToolbox IDE](#)
- [Build, Program, Debug Application](#)

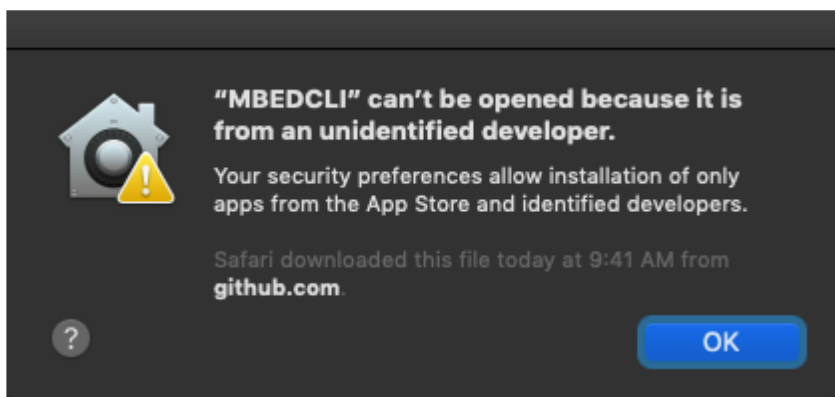
Install and Configure Mbed CLI

You should be familiar with Arm Mbed OS and the command line interface (CLI). Refer to the official Mbed OS instructions:

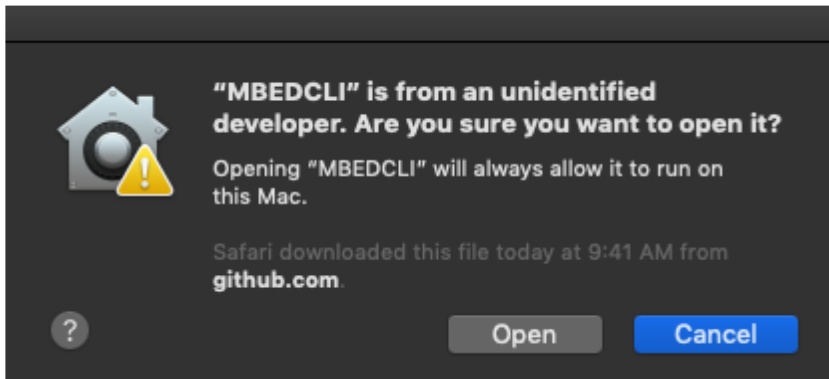
<https://os.mbed.com/docs/mbed-os/v5.11/tools/installation-and-setup.html>

For Windows and macOS, download and use the installer rather than manually installing it. The "see documentation" link on the Mbed webpage is for manual installation.

For macOS, you will get an error that it is from an unidentified developer.



If you get this message, right-click on the MBEDCLI application and select **Open**. Then, the message will look like the following:



Click **Open**.

Note You only have to do this once; macOS will remember the decision, and the app will open normally from then on

Install/Update PyOCD

In order for Mbed applications to work with ModusToolbox, make sure the Mbed Python packages are installed and updated to versions compatible with Cypress kits. See <https://github.com/mbedmicro/pyOCD#installing> for more details, including information about libusb installation.

For macOS, run the MBEDCLI application to start an mbed-capable terminal before installing pyocd. This ensures pyocd installs in the self-contained mbed environment. PyOCD is directly supported by the Python bundled with the MBEDCLI application.

Run the following command from a terminal window:

```
pyocd --version
```

The reply should be (or later):

```
0.16.1
```

If you don't have the correct version, run:

```
pip install -U pyocd
```

Configure Compilers

Install and configure the appropriate compilers for Mbed CLI. Refer to the Mbed OS for website supported compilers:

<https://os.mbed.com/docs/mbed-os/latest/tools/after-installation-configuring-mbed-cli.html>

To use the GCC 7.2.1 distribution bundled with ModusToolbox, run the `mbed config` command as appropriate for your operating system:

For Windows:

```
mbed config -G GCC_ARM_PATH %USERPROFILE%\ModusToolbox\tools_2.0\gcc-7.2.1\bin
```

For macOS:

```
mbed config -G GCC_ARM_PATH /Applications/ModusToolbox/tools_2.0/gcc-7.2.1/bin
```

For Linux:

```
mbed config -G GCC_ARM_PATH ~/ModusToolbox/tools_2.0/gcc-7.2.1/bin
```

To use another compiler, run the `mbed config` command and enter the appropriate path. The following are a few typical paths for compilers on Windows:

```
mbed config -G GCC_ARM_PATH "C:\Program Files (x86)\GNU Tools ARM Embedded\6 2017-q2-update\bin"
mbed config -G ARM_PATH "C:\Program Files (x86)\ARM Compiler_5.06u6"
mbed config -G ARMCC6_PATH "C:\Program Files\ARMCompiler6.11\bin"
mbed config -G IAR_PATH "C:\Program Files (x86)\IAR Systems\Embedded Workbench 7.5\arm"
```

Note IAR 8 is not compatible with Mbed OS 5.11.

To use the ARM v5/v6 compilers bundled with Keil MDK v5, use one of these commands as appropriate:

```
mbed config -G ARM_PATH "C:\Keil_v5\ARM\ARMCC"
mbed config -G ARMCC6_PATH "C:\Keil_v5\ARM\ARMCLANG\bin"
```

Switch Kit to DAPLink Mode

Use the `fw-loader` tool included with ModusToolbox software to upgrade the kit firmware using the following instructions. Refer also to the *KitProg3 User Guide* for more information. The tool is located in the following directory, by default:

```
<install_dir>/tools_2.0/fw-loader/bin/
```

Note For Windows 7 only, the Mbed serial port driver must be installed on your system with the connected kit. This driver should be installed with the Mbed CLI. If not, the driver can be found at: <https://os.mbed.com/handbook/Windows-serial-configuration>. Do not install this on Windows 10.

- Run the `fw-loader` tool from the command line to update the KitProg3 firmware:

```
fw-loader --update-kp3
```

- After updating the firmware, run this command to switch to DAPLink mode:

```
fw-loader --mode kp3-daplink
```

- Then, run the following from the command line:

```
pyocd list
```

The reply should indicate that a valid device is connected. For example:

```
#   Probe                                     Unique ID
-----
0   CY8CKIT-062-WIFI-BT [cy8c6xx7]  190013016c121817036c1218000000000000000002e127069
```

Note If you get an error message that `libusb` library was not found, make sure you have completed all the required steps described in <https://github.com/mbedmicro/pyOCD#installing>.

Create Application in Mbed CLI

Open an Mbed CLI prompt and create the directory for Mbed examples. For example:

```
mkdir mbed-examples
cd mbed-examples
```

Note On Windows, do **not** use a Cygwin terminal window, and make sure Cygwin is not in your Windows `PATH` system environment variable. Use the MINGW bash terminal instead.

Import the `mbed-os-example-blinky` example:

```
mbed import mbed-os-example-blinky
```

Switch to the example directory:

```
cd mbed-os-example-blinky
```

Compile Application in Mbed CLI (Optional)

Note This step is optional. It may be useful to confirm that the application builds successfully and produces a HEX file.

Compile the Mbed application for one of the supported target boards with one of the supported toolchains

```
mbed compile --target TARGET --toolchain TOOLCHAIN
```

where TARGET is one of:

```
CY8CKIT_062_WIFI_BT
CY8CPROTO_062_4343W
CY8CKIT_062_BLE
CY8CKIT_062_4343W
```

and TOOLCHAIN is one of:

```
GCC_ARM
ARM
IAR
```

For example:

```
mbed compile --target CY8CKIT_062_WIFI_BT --toolchain GCC_ARM
```

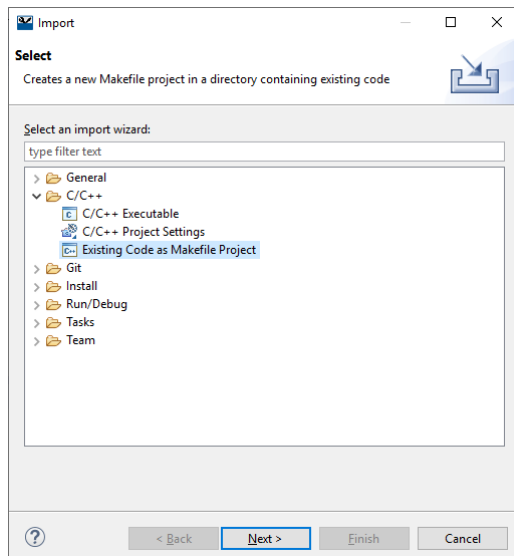
Export Application from Mbed CLI

To export the Mbed application for use in the ModusToolbox IDE, use the Mbed CLI `export` command. For example:

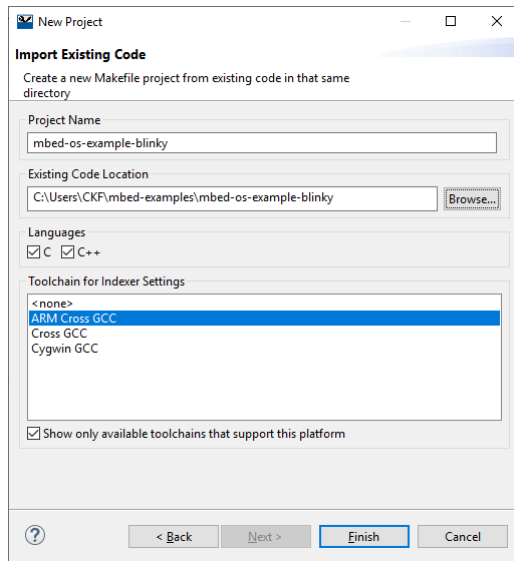
```
mbed export -i eclipse_gcc_arm -m CY8CKIT_062_WIFI_BT
```

Import Application into ModusToolbox IDE

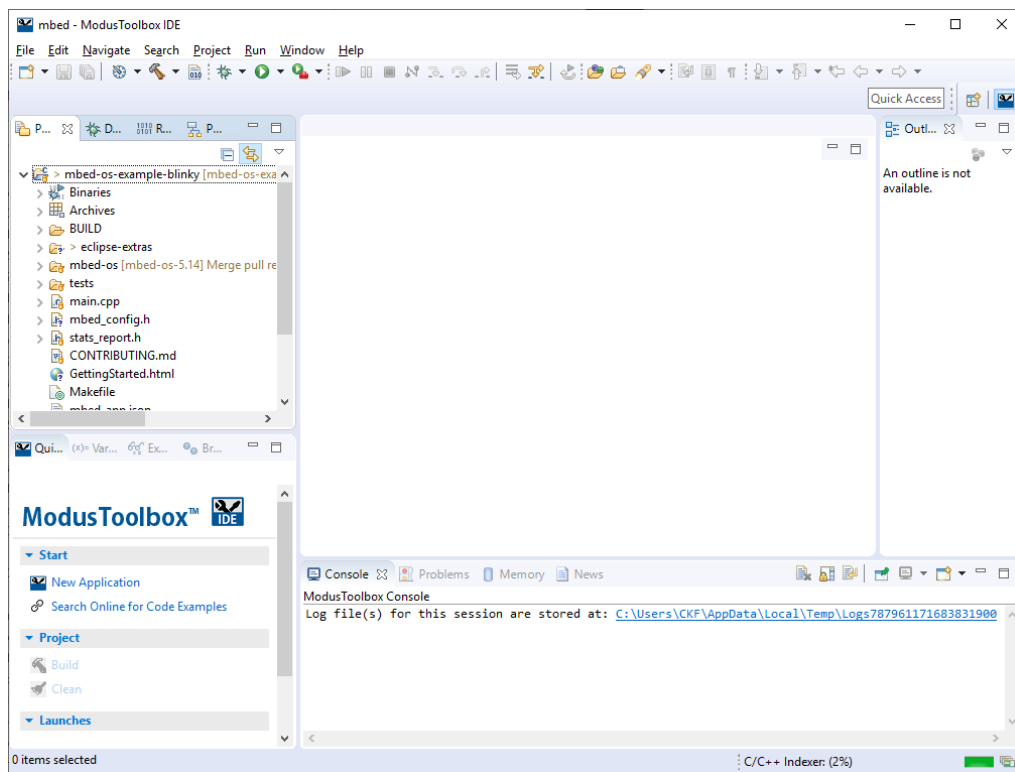
Use the standard Eclipse **Import** function to import the Mbed application. Expand the **C/C++** folder and select the **Existing Code as Makefile Project** option. Click **Next >**.



Click **Browse...** and navigate to the directory where you exported the application. Ensure the **Toolchain for Indexer Settings** is set to **ARM Cross GCC**. Do not change any other settings. Click **Finish**.



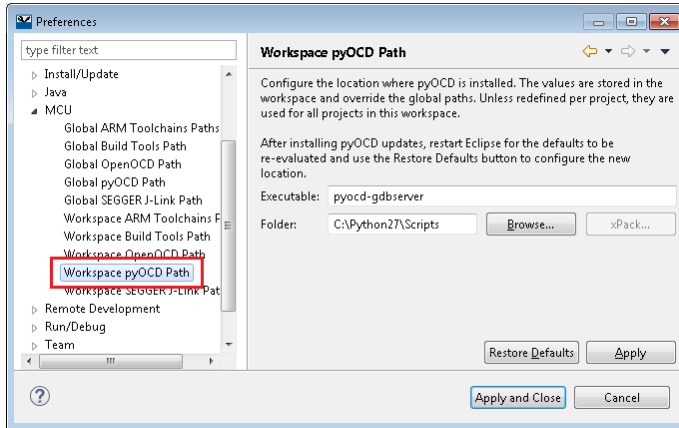
When the import finishes, the Mbed OS application will be shown in the [Project Explorer](#).



Configure ModusToolbox IDE

Define the Workspace pyOCD Path

Open the Preferences dialog, select **MCU > Workspace pyOCD Path**, and set the following workspace paths (adjust the path to the Scripts directory for your python/pyocd installation):



Note Your path folder may differ from these examples. Check the correct path folder using the `which pyocd-gdbserver` command.

Windows:

- Workspace pyOCD Path > Executable = pyocd-gdbserver
- Workspace pyOCD Path > Folder = C:\Python27\Scripts

macOS:

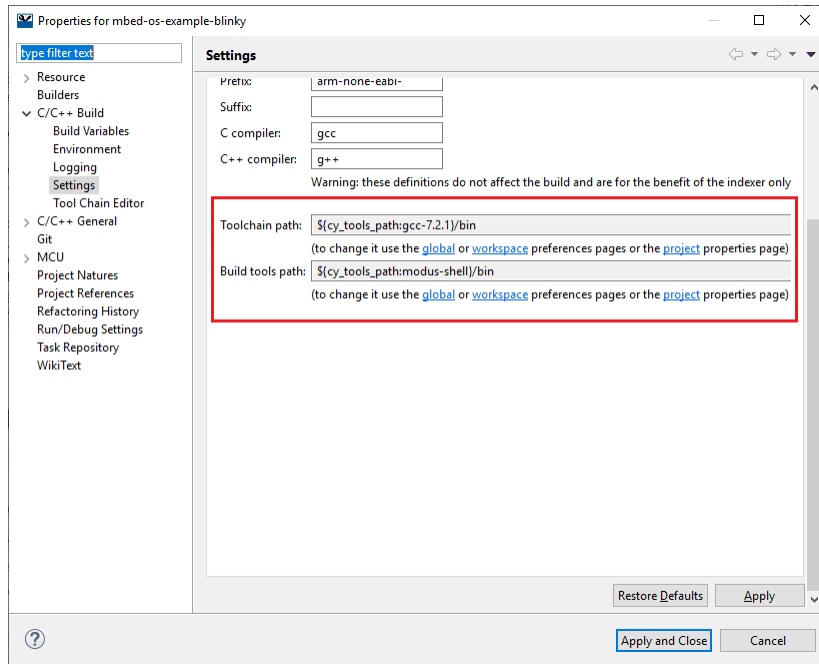
- Workspace pyOCD Path > Executable = pyocd-gdbserver
- Workspace pyOCD Path > Folder = /Applications/MBEDCLI.app/Contents/Resources/miniconda/bin

Linux:

- Workspace pyOCD Path > Executable = pyocd-gdbserver
- Workspace pyOCD Path > Folder = ~/.local/bin

Verify Properties Settings

Right-click on the project and select **Properties**. Navigate to **C/C++ Build > Settings**. Verify that the Toolchain path and Build Tools path have appropriate values. Click **Apply** and then click **Apply and Close**.



Note This step is needed due to a defect in Eclipse.

Build, Program, Debug Application

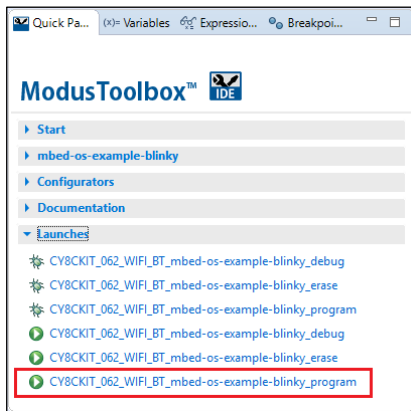
Build

Select **Project > Build** to build the project and generate the HEX file. When complete, the Console displays a message similar to the following:

```
link: mbed-os-example-blinky.elf
arm-none-eabi-objcopy -O binary mbed-os-example-blinky.elf mbed-os-example-blinky.bin
arm-none-eabi-objcopy -O ihex mbed-os-example-blinky.elf mbed-os-example-blinky.hex
```

Program

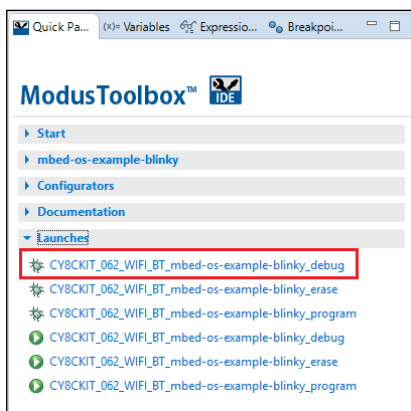
In the Quick Panel, under **Launches**, click the "CY8CKIT-062_WIFI_BT_mbed-os-example-blinky_program" link.



Messages will display in the Console. When programming is complete, the LED should blink red on the board. You may need to press the **Reset** button.

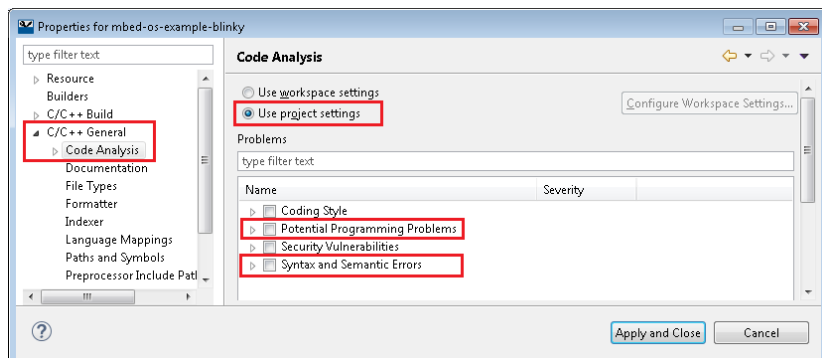
Debug

In the Quick Panel, under **Launches**, click the "CY8CKIT-062_WIFI_BT_mbed-os-example-blinky_debug" link.



The IDE will switch to debugging mode and will halt at the break, ready for debugging.

Note While debugging, you may see various errors in your code (*main.cpp* file), such as "function 'printf' could not be resolved." These are likely not real errors. They are caused by the import of the make file into Eclipse. To turn these errors off, go to **Project > Properties > C/C++ > Code Analysis**, and disable "Potential Programming Problems" and "Syntax and Semantic Errors."



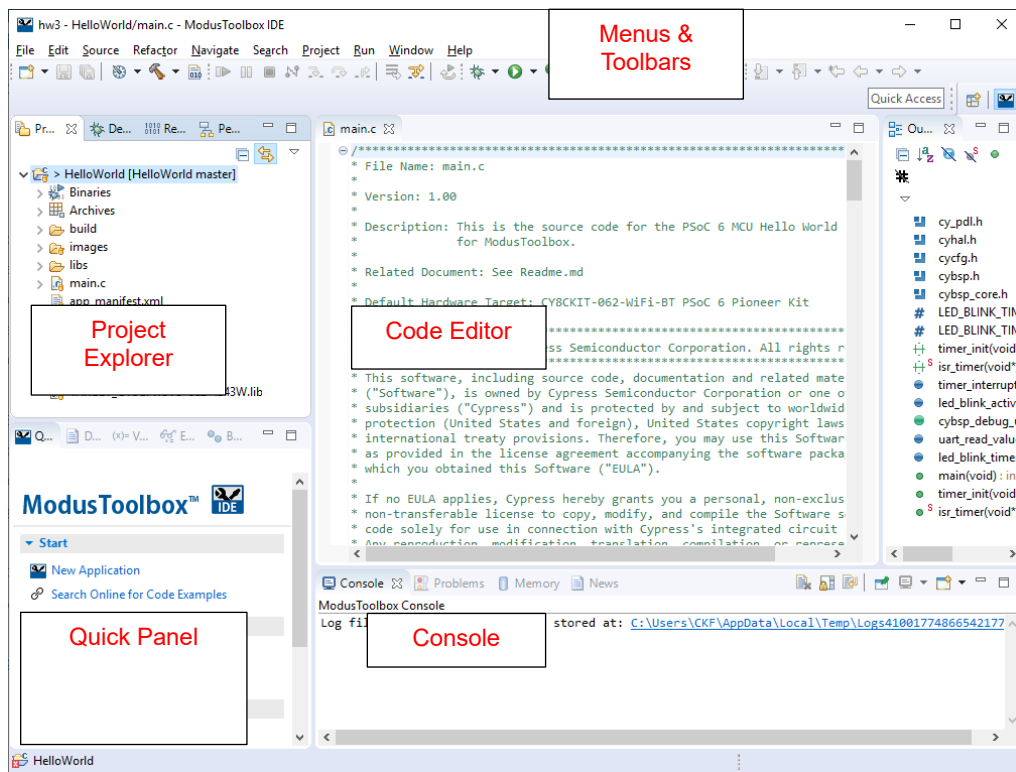
4. IDE Description



Overview

The ModusToolbox IDE is based on the Eclipse IDE. It uses several plugins, including the Eclipse C/C++ Development Tools (CDT) plugin. For more information about Eclipse, refer to the [Eclipse Workbench User Guide](#). Cypress also provides a document called the [Eclipse Survival Guide](#), which provides tips and hints for how to use the ModusToolbox IDE.

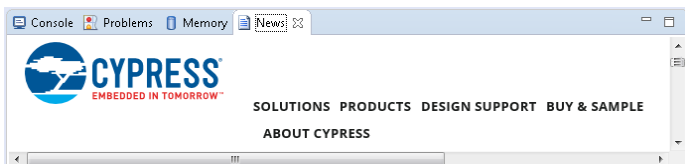
The IDE contains Eclipse standard menus and toolbars, plus various panes such as the Project Explorer, Code Editor, and Console. One difference from the standard Eclipse IDE is the "ModusToolbox Perspective." This perspective provides the "Quick Panel," a "News View," and adds tabs to the Project Explorer. "Perspective" is an Eclipse term for the initial set and layout of views in the IDE. The ModusToolbox IDE also provides a Welcome Page, which displays on first launch of the IDE for a given workspace.



Note If you switch to a different perspective, you can restore the ModusToolbox Perspective by clicking the ModusToolbox icon button in the upper right corner. You can also select **Perspective > Open Perspective > ModusToolbox** from the **Window** menu. To restore the ModusToolbox perspective to the original layout, select **Perspective > Reset Perspective** from the **Window** menu.

The following describe different parts of the IDE:

- Menus and Toolbars – Use the various menus and toolbars to access build/program/debug commands for your application. Many of these are covered in the [Eclipse Workbench User Guide](#).
- Project Explorer – Use the Project Explorer to find and open files in your application. See [Project Explorer](#) for more information.
- [Quick Panel](#) – Use this tab to access appropriate commands, based on what you select in the Project Explorer.
- News View – This is an Eclipse view in the ModusToolbox Perspective that displays a page of blog articles from cypress.com. This is located in the same panel as the Console and Problems tabs.

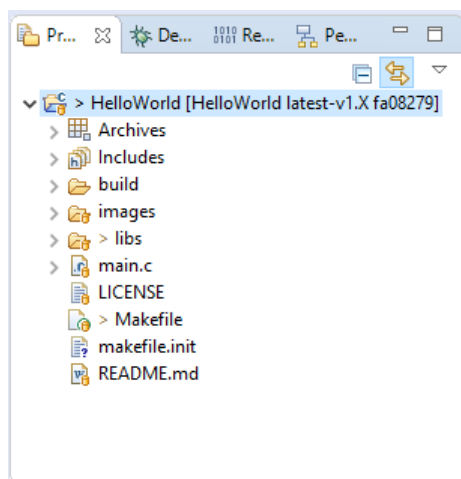


- Code Editor – Use the Code Editor to edit various source files in your application.

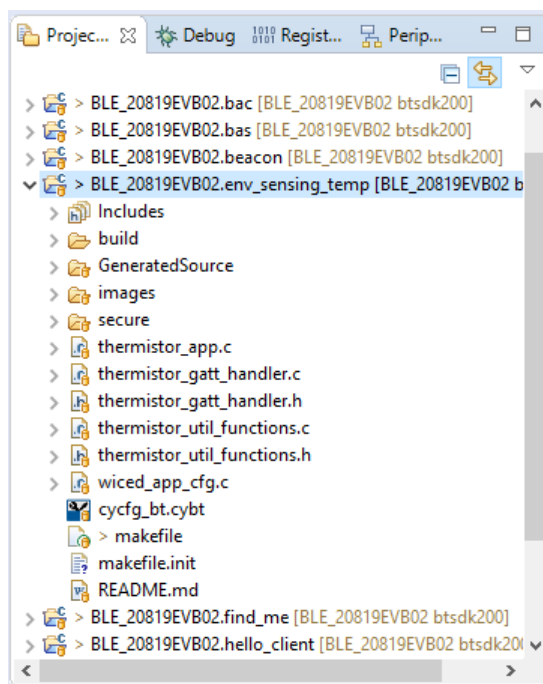
Project Explorer

In the ModusToolbox IDE, after creating an application, the Project Explorer contains one or more related project folders. The following images show a PSoC 6 MCU application and a WICED Bluetooth application. For applications imported from Mbed OS, see [Mbed OS to ModusToolbox Flow](#).

PSoC 6 MCU Application



WICED Bluetooth Application

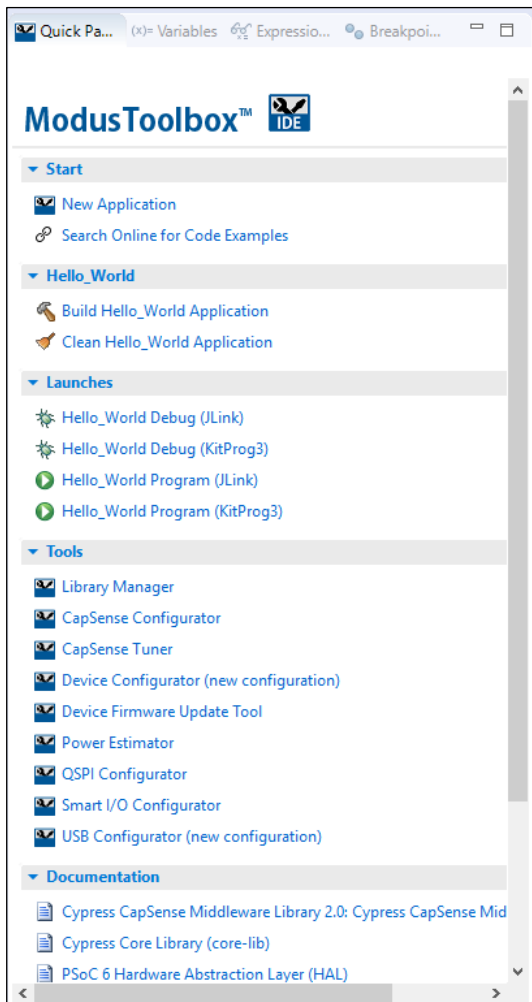


Both types of applications contain a similar project structure. Each contains the main application source code, and a Makefile. Note that PSoC 6 MCU applications contain a libs directory, which BT applications have a wiced_btsdk project with shared SDK, BSPs, and libraries for all Bluetooth applications in a workspace.

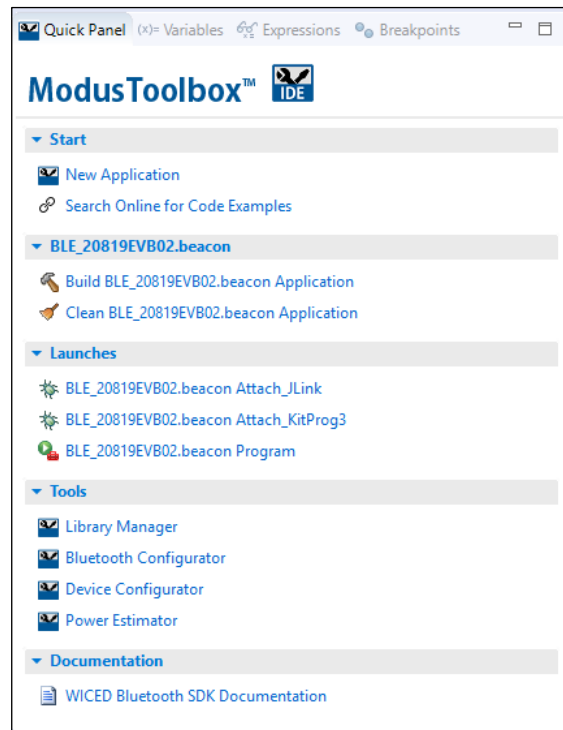
Quick Panel

As stated previously, the Quick Panel is part of the ModusToolbox Perspective. It provides quick access to commands and documentation based on what you have selected in the [Project Explorer](#).

PSoC 6 MCU Application



WICED Bluetooth Application



The Quick Panel contains links to various commands and documentation, organized as follows:

- **Start** – This contains the New Application link to create new applications, and a link to find Code Examples.
- **Selected <app-name>_project** – This contains different project-related links based on the project that is selected in the Project Explorer, as well as the type of application. Links here include: Build and Clean the application.
- **Launches** – This contains various Launch Configurations, based on the selected application project and device, which can be used to program the device and launch the debugger. This area is only populated if you have the top project in your application selected (<app-name>). For more information, see [Launch Configurations](#).
- **Tools** – This contains links to the various tools available for the selected project.
- **Documentation** – This may contain several documents in HTML format, which are included as part of the chosen BSP.

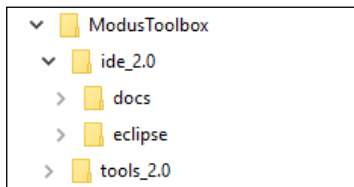
5. Installation Description



The installer provides the central core of ModusToolbox software. It contains configuration tools, build system infrastructure, and the IDE. You may use one or more of these tools in any environment you prefer.

Directory Structure

Refer to the [ModusToolbox Installation Guide](#) for information about installing ModusToolbox. Once it is installed, the various ModusToolbox top-level directories are organized as follows:



These directories contain the following files and folders:

- **ide_2.0**
 - **docs** – This is the top-level documentation directory. It contains various top-level documents and an html file with links to documents provided as part of ModusToolbox. See [Documentation](#) for more information.
 - **eclipse (or ModusToolbox.app on macOS)** – This contains the IDE. See [IDE Description](#).
- **tools_2.0** – This contains all the various tools and scripts provided as part of ModusToolbox. See [Tools](#) for more information.

Documentation

The `/ide_2.0/docs` directory contains top-level documents and an HTML document with links to all the documents included in the installation and on the web.

Release Notes

For the 2.0 release, the release notes document is for all of the ModusToolbox software included in the installation.

Top-Level Documents

This folder contains the ModusToolbox IDE Quick Start Guide and this user guide. These guides cover different aspects of using the IDE.

Document Index Page

The *doc_landing.html* file provides links to all the documents included in the installation and on the web. This file is available from the IDE **Help** menu.

ModusToolbox™ 2.0 Documentation

This page provides brief descriptions and links to various types of documentation included as part the ModusToolbox software.

Note: Many of these documents are also provided online at the [ModusToolbox website](#). Also, some of the documents online might be more current than versions installed on disk.

Getting Started Documents

This section contains general documents to install ModusToolbox software, use the IDE, learn tips for using ModusToolbox in Eclipse, and porting applications from other Cypress IDEs.

Name	Description
ModusToolbox Installation Guide	This document is available online only. It describes how to install the ModusToolbox software on Windows, Linux, and macOS.
ModusToolbox Release Notes	Features and Known Issues for the ModusToolbox 2.0 release.
ModusToolbox IDE Quick Start Guide	Step-by-step guide to create and build starter applications using the IDE.
ModusToolbox IDE User Guide	Guide covering more details about the ModusToolbox IDE and software features.
Eclipse Survival Guide	This document is also online only. It offers tips on how the ModusToolbox IDE works in the Eclipse environment.
Running ModusToolbox from the Command Line	Guide for using the command-line make for ModusToolbox applications.
EULA	End user license agreement; provided on disk as part of installation.

Configurator and Tool Documents

These documents are located in the "tools" directory in each individual configurator and tool "docs" subfolder.

Name	Description
Device Configurator Guide	Covers how to enable and configure platform peripherals, such as clocks and pins, as well as standard MCU peripherals that do not require their own tool.

Tools

The */tools_2.0* folder contains the following tools:

- ▼ tools_2.0
 - > bt-configurator
 - > capsense-configurator
 - > cfg-backend-cli
 - > cymcuelftool-1.0
 - > cype-tool
 - > device-configurator
 - > dfuh-tool
 - > driver_media
 - > fw-loader
 - > gcc-7.2.1
 - > jre-1.0
 - > library-manager
 - make
 - > modus-shell
 - > openocd
 - > project-creator
 - > qspi-configurator
 - > seglcd-configurator
 - > smartio-configurator
 - > usbdev-configurator

- Configurators – There are several configurators used to update various settings for different peripherals. See [Use Configurators](#).
- cfg-backend-cli – This contains backend support files used by the system. You do **not** need to interact with this folder.
- cymcuelftool-1.0 – This tool is used to manipulate Elf files. Refer to the *CyMCUElfTool User Guide* located in the tool's doc folder.
- cype-tool – This is the Low Power Estimator tool.
- dfuh-tool – This tool is used to communicate with and update firmware on a PSoC 6 MCU that has already been programmed with an application that includes device firmware update capability.
- driver_media – This folder contains WICED board drivers.
- fw-loader-2.2 – This is the Firmware Loader tool used to update firmware on the programmer/debugger device on PSoC 6 MCU kits. See [KitProg Firmware Loader](#).
- gcc-7.2.1 – ModusToolbox software includes GCC version 7.2.1 as the preferred toolchain. See <https://www.gnu.org/software/gcc/> for information.
- jre-1.0 – This folder contains the Java Runtime Environment version provided as part of the tool. This is used by the IDE and the backend. See <https://www.java.com> for more information.
- library-manager – This is the Library Manager tool. See [Update BSPs and Libraries](#) for more details.
- make – This folder contains scripts for the build system.
- modus-shell – This folder contains various helper utilities used by the system. You do **not** need to interact with this folder.
- openocd – This contains the version of the Open On-Chip Debugger used by ModusToolbox to program various boards. For more information, refer to the *Cypress Programmer User Guide*.
- project-creator – Tool used to create projects when you do not want to use the IDE.

6. Configure Applications



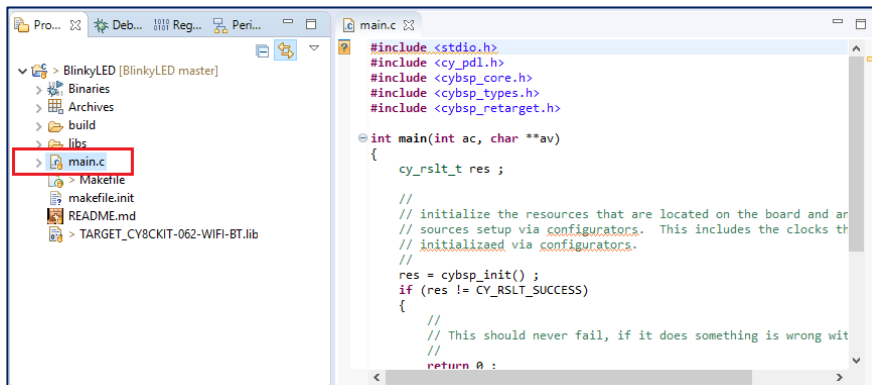
This chapter covers how to make various changes to your application. It includes:

- [Modify Code](#)
- [Use Configurators](#)
- [Use Tools](#)
- [Rename Application](#)

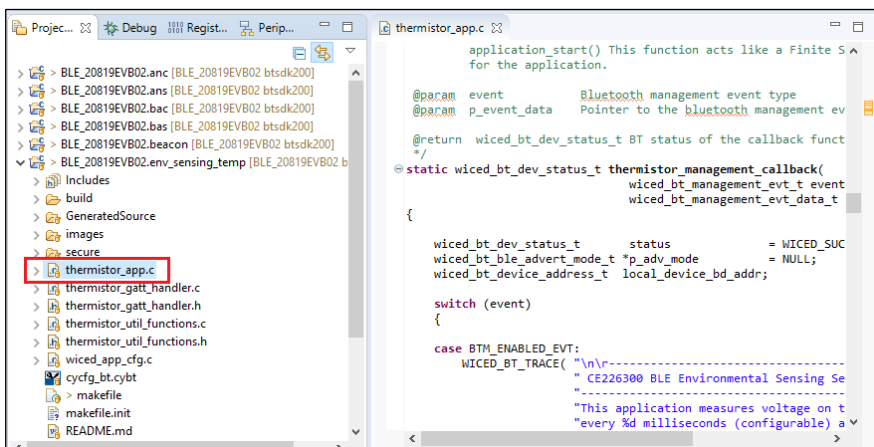
Modify Code

Most starter applications work as they are, and there is no need to add or modify code. However, if you want to update and change the starter application to do something else, or if you are developing your own application, open the appropriate file in the code editor.

- **PSoC 6 MCU:** In the Project Explorer, double-click the *main.c* file.



- **WICED Bluetooth:** In the Project Explorer, expand the <app-name> project folder and double-click the application <app-name>.c file.



As you type into the file, an asterisk (*) will appear in the file's tab to indicate changes were made. The **Save/Save As** commands will also become available to select.

Use Configurators

ModusToolbox software provides graphical applications called configurators that make it easier to configure a hardware block. For example, instead of having to search through all the documentation to configure a serial communication block as a UART with a desired configuration, open the appropriate configurator to set the baud rate, parity, stop bits, etc. Upon saving the hardware configuration, the tool generates the C code to initialize the hardware with the desired configuration.

Many configurators do not apply to all types of projects. So, the available configurators depend on the project/application you have selected in the Project Explorer. Configurators are independent of each other, but they can be used together to provide flexible configuration options. They can be used stand alone, in conjunction with other configurators, or within a complete IDE. Everything is bundled together as part of the installation. Each configurator provides a separate guide, available from the configurator's **Help** menu.

Configurators perform tasks such as:

- Displaying a user interface for editing parameters
- Setting up connections such as pins and clocks for a peripheral
- Generating code to configure middleware

Configurators store configuration data in an XML data file that provides the desired configuration. Each configurator has a "command line" mode that can regenerate source based on the XML data file. Configurators are divided into two types: Board Support Package (BSP) Configurators and Library Configurators.

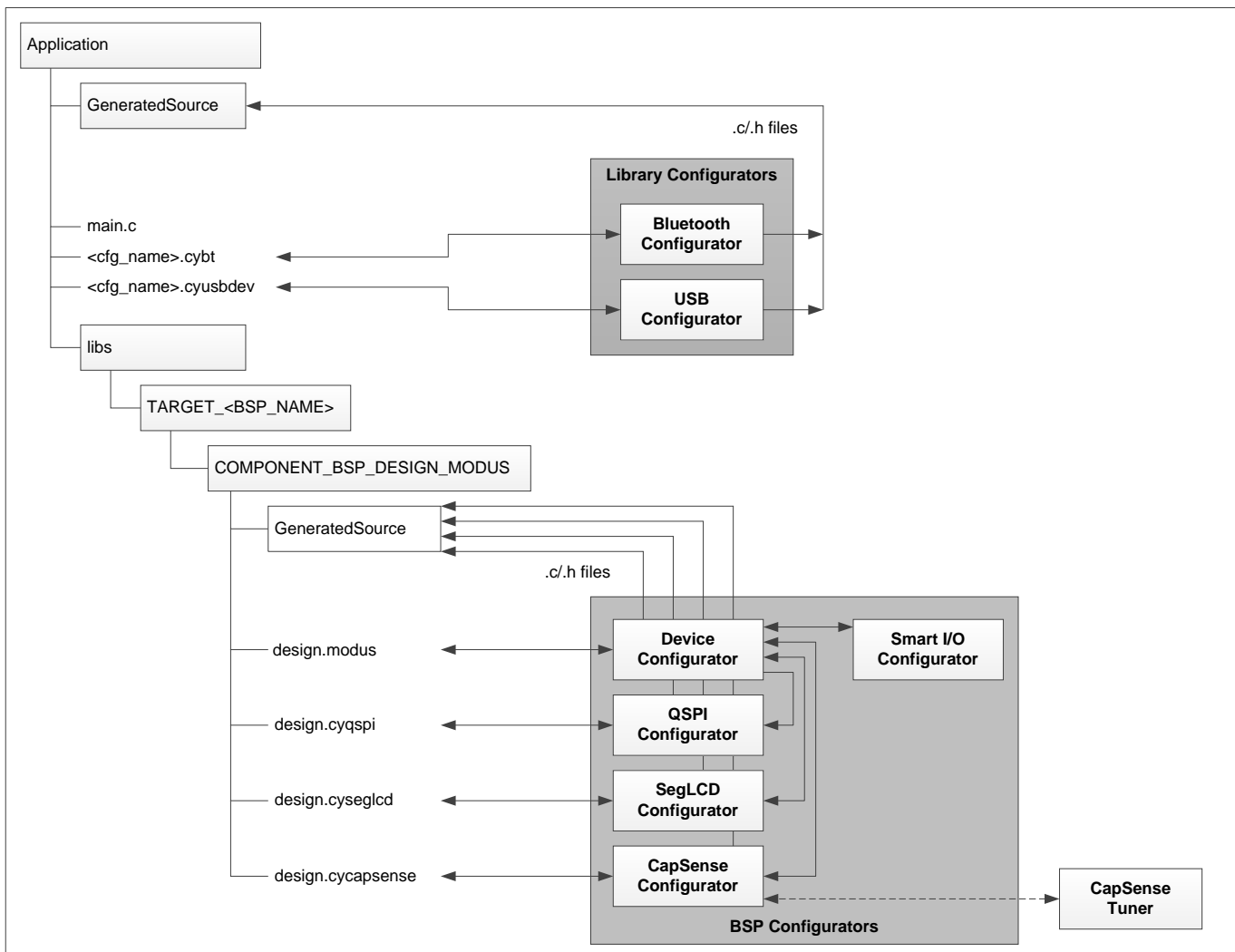
BSP Configurators include:

- **Device Configurator:** Set up the system (platform) functions such as pins, interrupts, clocks, and DMA, as well as the basic peripherals, including UART, Timer, etc.
- **CapSense Configurator:** Configure CapSense hardware, and generate the required firmware. This includes tasks such as mapping pins to sensors and how the sensors are scanned.
- **QSPI Configurator:** Configure external memory and generate the required firmware. This includes defining and configuring what external memories are being communicated with.
- **Smart I/O™ Configurator:** Configure the Smart I/O. This includes defining and configuring what external memories are being communicated with.
- **SegLCD Configurator:** Configure LCD displays. This configuration defines a matrix Seg LCD connection and allows you to setup the connections and easily write to the display.

Library configurators include:

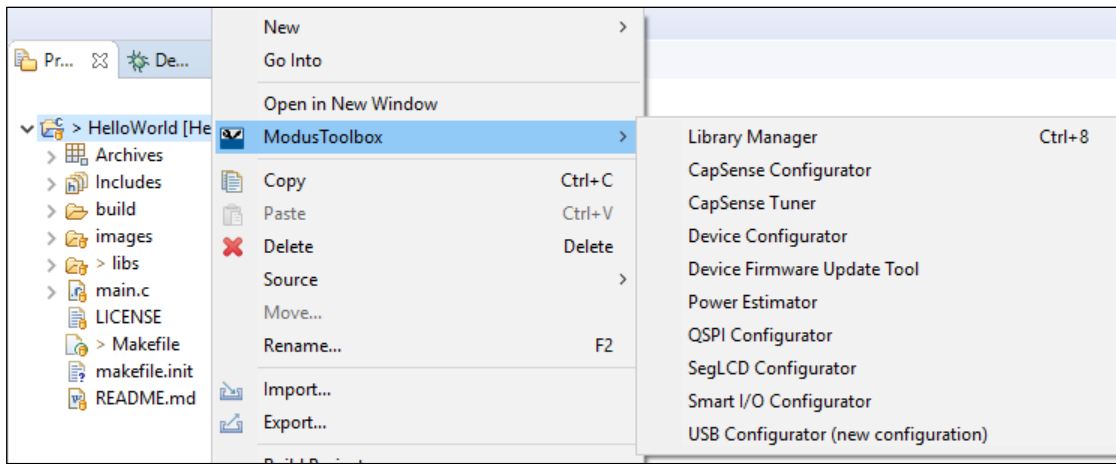
- **Bluetooth Configurator:** Configure Bluetooth settings. This includes options for specifying what services and profiles to use and what features to offer by creating SDP and/or GATT databases in generated code. This configurator supports both PSoC MCU and WICED Bluetooth applications.
- **USB Configurator:** Configure USB settings and generate the required firmware. This includes options for defining the 'Device' Descriptor and Settings.

The following diagram shows a high-level view of the Configurators in a typical application. For a more comprehensive description of the configurators, refer to the *ModusToolbox Software Overview* document.



Launching Configurators from the IDE

To launch a configurator from the ModusToolbox IDE, right-click on the `<app-name>` project in the Project Explorer, select **ModusToolbox**, and then select the appropriate configurator.



Note You can also launch available configurators from links in the Quick Panel.

Depending on the enabled resources in your application, there may be several configurators available to launch.

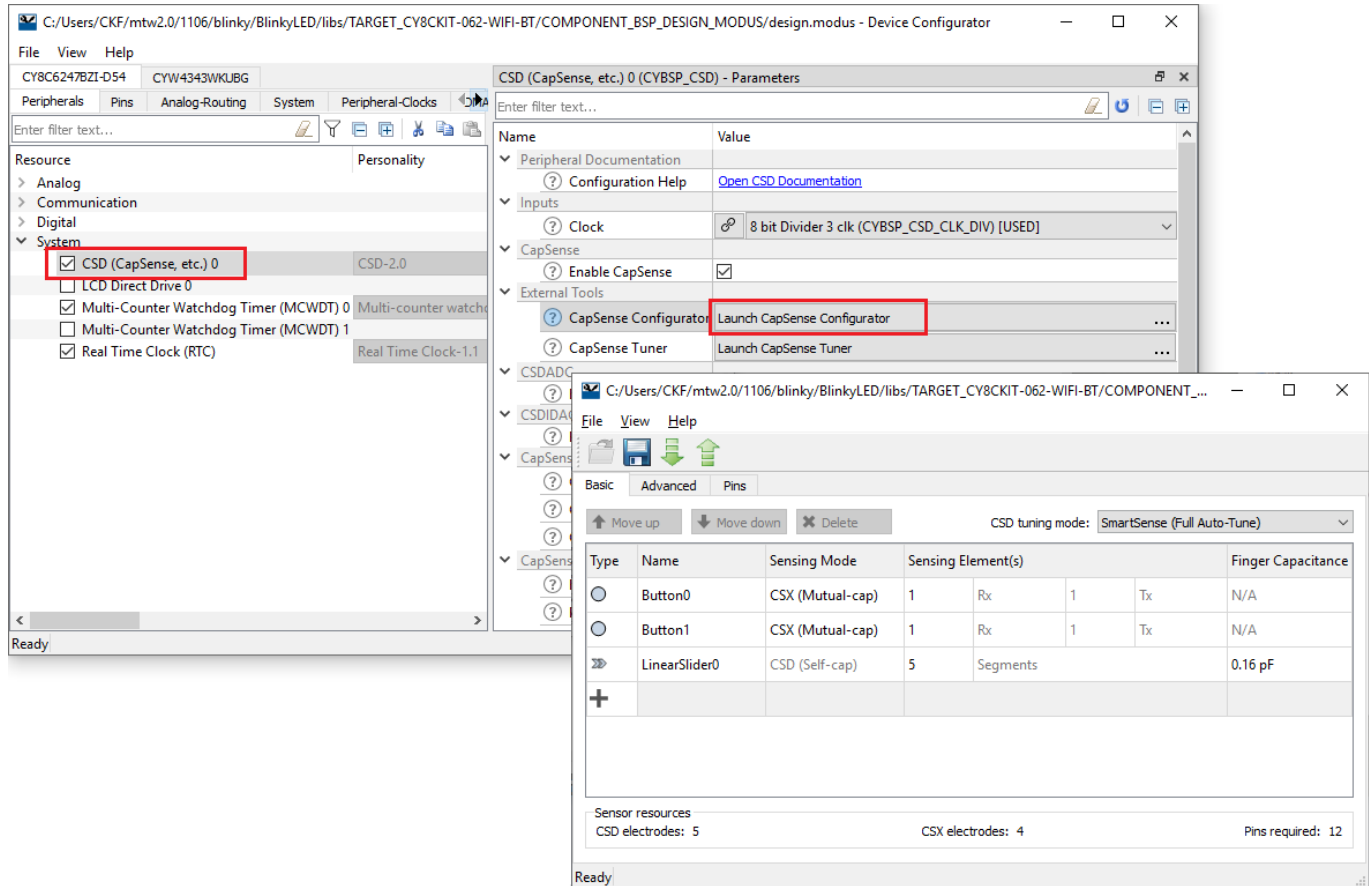
- If you launch the Device Configurator from the IDE, you are opening the project's *design.modus* file, which is responsible for holding all of the BSP configuration information. It contains the following:
 - Selected device
 - Resource parameters
 - Constraints

- If you launch any of the other configurators from the IDE, they will open using that configurator's configuration file (*design.cycapsense*, *design.cysegLCD*, etc.). These files are specific to the given resource, and they may rely on configuration data from the *design.modus* file.

Launching BSP Configurators from Device Configurator

You can launch [BSP Configurators](#) from the Device Configurator; however, you must launch [Library Configurators](#) independently. All configurators can be launched independently, as well as from the command line. Refer to the individual configurator guides for more information. These guides are available from the configurators' **Help** menu.

To launch a BSP Configurator from the Device Configurator, enable and/or select the desired resource under **Peripherals > Resource**, then click the **[Launch . . .]** button under **Parameters**.



In this example, the CapSense Configurator opens in a separate window, and it contains various configuration information related to the application.

Launching Configurators without the IDE

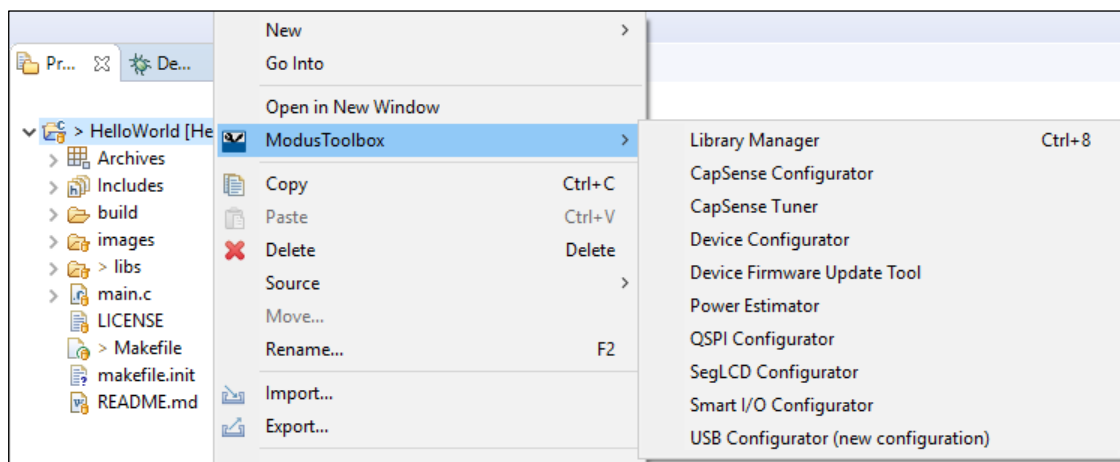
To launch any Configurator without using the IDE, navigate to the desired configurator's installation directory, and run the executable. Some configurators will open without any configuration information. You will need to open a configuration file or create a new one. Others will open with a default configuration file that needs to be saved.

Use Tools

In addition to the configurators, there are several tools, including Library Manager, BTSpy, ClientControl, Power Estimator, and Device Firmware Update (DFU) Host Tool. Many tools do not apply to all types of projects. So, the available tools depend on the project/application you have selected in the Project Explorer.

Launching Tools from the IDE

To launch a tool from the ModusToolbox IDE, right-click on the *<app-name>* project in the Project Explorer, select **ModusToolbox**, and then select the appropriate tool.



Note You can also launch available configurators from links in the Quick Panel.

Library Manager

The Library Manager allows you to select which Board Support Package (BSP) and version should be used by default when building a ModusToolbox application. It also allows you to add and remove libraries, as well as change their versions.

For more information about how to use this tool, refer to the *Library Manager User Guide*, available from the tool's **Help** button.

BTSpy and ClientControl

BTSpy is a trace utility that can be used in the WICED Bluetooth applications to view protocol and generic trace messages from the embedded device. The tool listens on the UDP port 9876 and can receive specially formatted message from another application on the same or different system.

BTSpy can be used in conjunction with ClientControl to receive, decode and display protocol application and stack trace messages. ClientControl communicates with the embedded app to perform various functionalities, tests, exercising features, etc.

DFU Host Tool

The Device Firmware Update (DFU) Host tool is a stand-alone program used to communicate with a PSoC 6 MCU that has already been programmed with an application that includes device firmware update capability. For more information, refer to the *Device Firmware Update Host Tool* guide, available from the tool's **Help** menu.

Power Estimator

The Power Estimator tool provides an estimate of power consumed by a target device (also called platform). This is a stand-alone tool included with the ModusToolbox software. Currently, it can only be used with the following devices:

- CYW920819EVB_02 (BTSDK)
- CY8CKIT-062S2-43012 (Mbed OS)

For more information, refer to the *Power Estimator User Guide*, available from the tool's **Help** menu.

Rename Application

The ModusToolbox IDE uses the standard Eclipse rename functionality. That is, right-click on the project and select **Rename**. If you use the rename feature, you will need to update your application's launch configurations. Open a terminal window in the application's folder, and type the following command:

```
make eclipse CY_IDE_PRJNAME=<New Eclipse project name>
```

For more information about make commands, refer to the *Running ModusToolbox from the Command Line* document, available from the IDE's **Help** menu.

7. Build Applications



This chapter covers various aspects of building applications. Building applications is not specifically required, because building is performed as part of the [programming and debugging process](#). However, if you are running the ModusToolbox IDE without any hardware attached, you may wish to build your application to ensure all the code is correct. If you changed code in one of your projects, you may wish to build just that project.

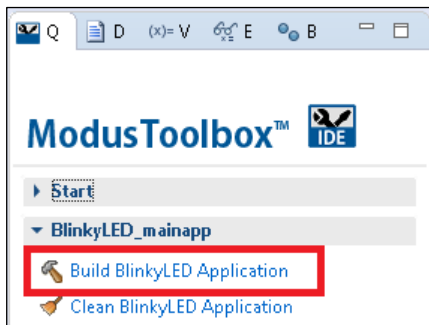
- [Build with Make](#)
- [Build with IDE](#)
- [Generated ELF Files](#)
- [Enable HEX File Generation](#)

Build with Make

You can build applications from the command line using Make. Refer to the *Running ModusToolbox from the Command Line* document located in the `<install>/ide_2.0/docs` directory for more information.

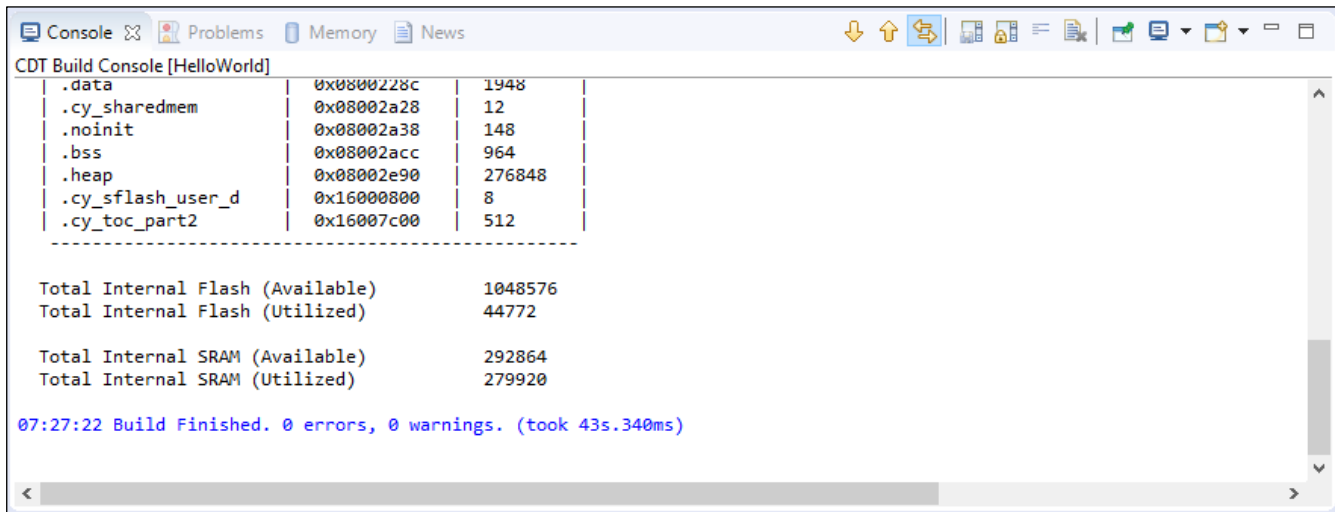
Build with IDE

After loading an application, it is best to first build everything to generate the necessary files. Click on a project in the Project Explorer. Then in the **Quick Panel**, click the "Build <app-name> Application" link.



Building an application will typically allow IntelliSense to work better. It may also be useful to right-click on a project and select **Index > Rebuild** to allow IntelliSense to find references.

Messages display in the Console, indicating whether the build was successful or not.



```


CDT Build Console [HelloWorld]
-----
| .data                | 0x0800228c | 1948 |
| .cy_sharedmem        | 0x08002a28 | 12   |
| .noinit              | 0x08002a38 | 148  |
| .bss                 | 0x08002acc | 964  |
| .heap                | 0x08002e90 | 276848 |
| .cy_sflash_user_d    | 0x16000800 | 8    |
| .cy_toc_part2        | 0x16007c00 | 512  |
|-----|-----|-----|

Total Internal Flash (Available)    1048576
Total Internal Flash (Utilized)     44772

Total Internal SRAM (Available)     292864
Total Internal SRAM (Utilized)      279920

07:27:22 Build Finished. 0 errors, 0 warnings. (took 43s.340ms)
    
```

Note Be aware that there are several Console views available.

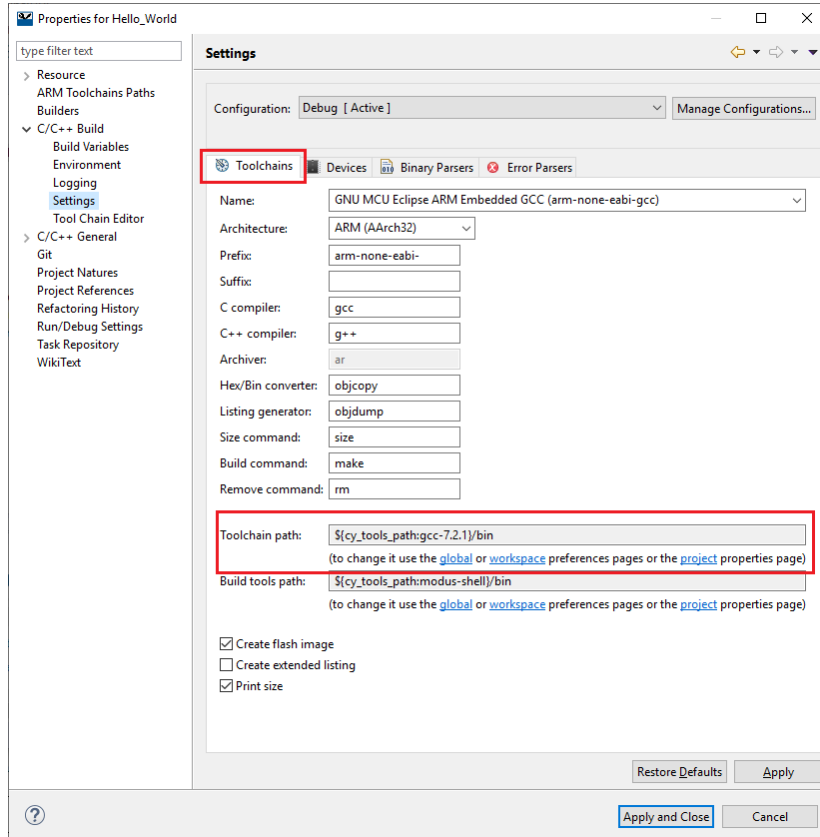
You can access the different views using the **Display Selected Console**  button (or the pull-down arrow). The Global Build Console is the only way to see all of the results in one place. If you just have the standard Console open, it resets every time a new project in the application starts building. You won't see any errors if they are not on the final project that gets built.

For subsequent updates, you can build one or more projects using the right-click menu options. Any projects that are dependent on the project being built will also be built. The ModusToolbox IDE supports all the usual application and build options available for the native Eclipse IDE.

Note When the active build configuration is changed it affects only the selected project. The active build configuration for any dependent projects is specified separately for each project. Therefore, if you want to use the same configuration for all projects (for example, Debug or Release), it must be set for each project. It is possible to select multiple projects at once from the Project Explorer and then select the active configuration.

GCC Version

ModusToolbox software includes GCC version 7.2.1 as the preferred toolchain to use with the ModusToolbox IDE. If you have a different version of GCC you prefer, update the project properties to point to the appropriate toolchain folder. Open the project Properties dialog, and click the **Toolchains** tab, and then click the appropriate link to update global or workspace preferences, or project properties.



8. Program and Debug



Programming and debugging is native to your chosen development environment. Cypress devices are supported in the major program and development solutions. Primarily, this means J-Link and OpenOCD. These solutions provide for programming flash within a device and provide a GDB server for debugging.

This chapter covers various topics related to building and debugging using the ModusToolbox IDE for PSoC 6 devices and WICED Bluetooth devices.

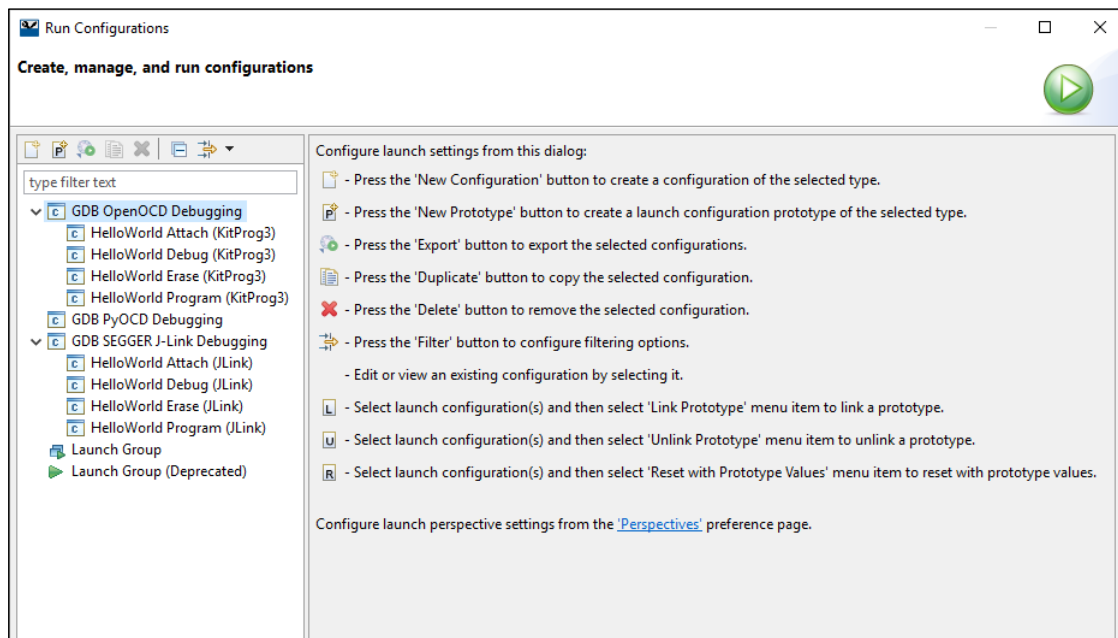
- [PSoC 6 Programming/Debugging](#)
 - [Launch Configurations](#)
 - [Attach to Running PSoC 6 Target](#)
 - [Programming eFuse](#)
 - [Debug Connection Options](#)
 - [Select Specific CMSIS-DAP Device](#)
 - [Select Specific J-Link Device](#)
 - [KitProg Firmware Loader](#)
- [Bluetooth Programming/Debugging](#)
 - [Launch Configurations](#)
 - [Debug Settings](#)
- [Mbed OS Programming/Debugging](#)
 - [Launch Configurations](#)
 - [Change PyOCD Message Color](#)
 - [Enable Peripherals View](#)
 - [Attach to Running Target \(PyOCD\)](#)

PSoC 6 Programming/Debugging

Launch Configurations

The flow for programming and debugging is similar for all devices. The ModusToolbox IDE contains several Launch Configurations that control various settings for programming the devices and launching the debugger. Depending on the kit and type of applications you are using, there are various Launch Configurations available.

There are two sets of configurations: one for KitProg3 (included on-board on most Cypress PSoC 6 based kits) and another for J-Link. These are shown in the Run/Debug Configurations dialog, similar to the following.



You can open these dialogs from the **Run** menu or by selecting the down arrow ▼ next to the **Run** and **Debug** commands.

These configurations include the application name and protocol, for example:

CapSenseSlider Program (KitProg3)

CapSenseSlider Debug (J-Link)

Note KitProg3 configurations may not work if a J-Link probe is attached to the kit.

When an application is created, the tool generates the following launch configurations for both KitProg3 and J-Link. Some items display in the **Quick Panel**, and some are in the Run/Debug Configurations dialog only.

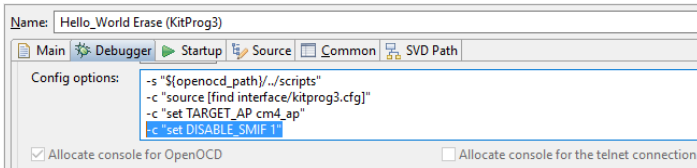
- **Attach:** This launch configuration starts a Cortex-M4 debugging session attaching to a running PSoC 6 target without programming or reset.
- **Debug:** This launch configuration builds the entire application on both cores, programs all the device's memories, and then starts a Cortex-M4 debugging session.
- **Erase:** This launch configuration erases all internal memories.
- **Program:** This launch configuration builds the entire application on both cores, programs all the device's memories, and then runs the program.

Erasing external memory

To erase external memory, you must modify the "Erase" launch configuration options on the **Debugger** tab as follows:

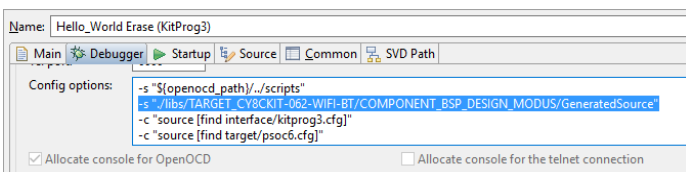
For the **CY8CKIT-064-S2-4343W** and **CY8CPROTO-064-SB** kits, remove the entry to disable SMIF:

```
-c "set DISABLE_SMIF 1"
```



For **all other kits**, add the path to the GeneratedSource folder (after `-s "${openocd_path}/../scripts"`):

```
-s ".libs/<TARGET_NAME>/COMPONENT_BSP_DESIGN_MODUS/GeneratedSource"
```

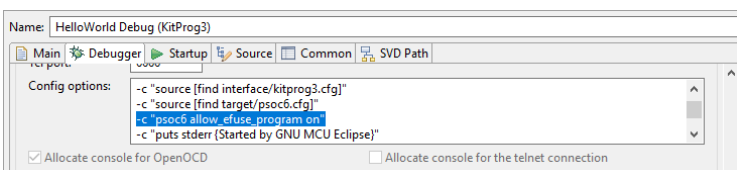


Programming eFuse

PSoc 6 MCUs contain electronic fuses (eFuses), which are one-time programmable. They store device specific information, protection settings and customer data.

By default eFuses are not programmed even if they are present in the programming file. To enable eFuse programming, add the following command for the appropriate Debug Configuration, under the **Debugger** tab in the **Config options** field (after `-c "source [find target/psoc6.cfg]"`):

```
-c "psoc6 allow_efuse_program on"
```



Warning Because blowing an eFuse is an irreversible process, Cypress recommends programming only in mass production programming under controlled factory conditions and not prototyping stages. Programming fuses requires the associated I/O supply to be at a specific level: the device VDDIO0 (or VDDIO if only one VDDIO is present in the package) supply should be set to 2.5 V (±5%).

Debug Connection Options

By default, a typical PSoC 6 application created in the ModusToolbox IDE includes [launch configurations](#) set up for two probes: Cypress KitProg3 (built into Cypress kits) and Segger J-Link.

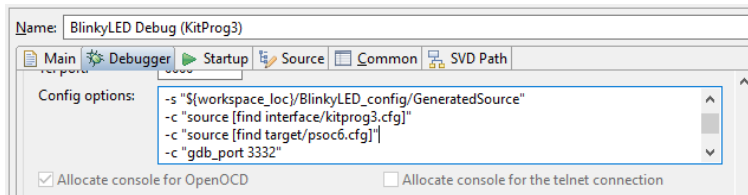
Communication Firmware	Debug Connection	More Information
Cypress-provided KitProg3	OpenOCD via CMSIS-DAP	https://www.cypress.com/products/psoc-programming-solutions
SEGGER-provided J-Link DLL	SEGGER J-Link	SEGGER J-Link

Select Specific CMSIS-DAP Device

If there are two or more CMSIS-DAP devices connected to your computer, the first detected device will be used by default. KitProg3 supports both CMSIS-DAP modes – HID and BULK. BULK devices are selected first, then HID devices. You can specify a CMSIS-DAP device by:

- VID and PID of the USB device
- Serial Number of the USB device
- VID, PID, and Serial Number of the USB device

To do this, you must add a specific command for the appropriate Debug Configuration, under the **Debugger** tab in the **Config options** field (after `-c "source [find interface/kitprog3.cfg]"`)



Selecting by VID and PID

Use OS-specific tools to determine the VID and PID of connected devices. For example, on Windows, use the Device Manager. Use the `"cmsis_dap_vid_pid"` command to select a CMSIS-DAP device with a specific VID and PID. If there are two or more devices with the same specified VID/PID pair, OpenOCD uses the first detected device from the passed list.

- To specify KitProg3 in CMSIS-DAP BULK mode with VID = 0x04B4 and PID = 0xF155:

```
cmsis_dap_vid_pid 0x04B4 0xF155
```

- To specify KitProg3 in CMSIS-DAP HID mode with VID = 0x04B4 and PID = 0xF154:

```
cmsis_dap_vid_pid 0x04B4 0xF154
```

- To specify any (HID or BULK) connected KitProg3 device:

```
cmsis_dap_vid_pid 0x04B4 0xF154 0x04B4 0xF155
```

Selecting by Serial Number

There should not be more than one device with the same serial number. Use this method if you want to use only one specific device. Use OS-specific tools to determine the Serial Number of connected devices. You can also use the `fw-loader` utility with `--device-list` option. See [KitProg Firmware Loader](#).

Use the "cmsis_dap_serial" command to select a CMSIS-DAP device with a specific Serial Number.

- To specify a CMSIS-DAP device with Serial Number = 0B0B0F9701047400 the following command is used:

```
cmsis_dap_serial 0B0B0F9701047400
```

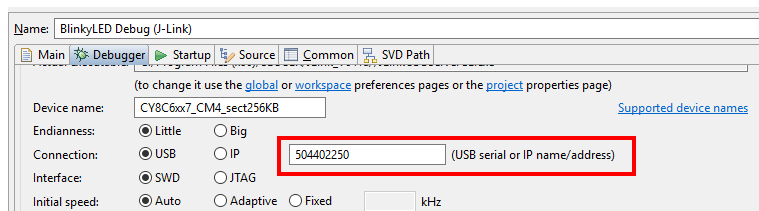
Selecting by both VID/PID and Serial Number

You can use both commands together in any order. For example:

```
cmsis_dap_vid_pid 04B4 F155 cmsis_dap_serial 0B0B0F9701047400
```

Select Specific J-Link Device

If there are two or more J-Link devices connected to your computer, the first detected device is used by default. You can select a specific J-link device by setting the serial number for the appropriate Debug Configuration, under the **Debugger** tab:



KitProg Firmware Loader

The PSoC 6 MCU kits include onboard programmer/debug firmware, called KitProg. The CY8CPROTO-062-4343W kit has KitProg3 by default. However, the CY8CKIT-062-BLE and CY8CKIT-062-WIFI-BT kits come with KitProg2 firmware installed, which does not work with the ModusToolbox software. **You must update to KitProg3.** KitProg3 provides the CMSIS-DAP (HID) protocol and the CMSIS-DAP (Bulk) protocol, which is up to ~2.5 times faster. Both modes can be used via OpenOCD.

ModusToolbox software includes a command-line tool "fw-loader" to update Cypress kits and switch the KitProg firmware from KitProg2 to KitProg3, and back. The following is the default installation directory of the tool on Windows:

```
<user_home>\ModusToolbox \tools_<version>\fw-loader\bin\
```

For other operating systems, the installation directory will vary, based on where the software was installed.

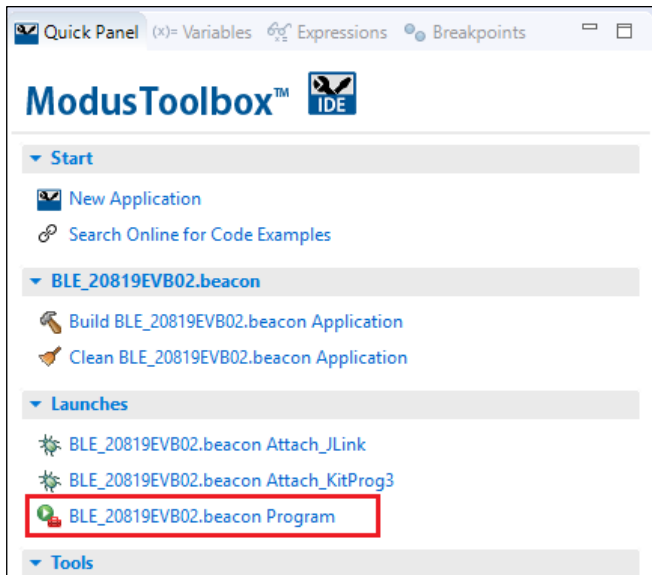
Use the fw-loader tool to update the KitProg firmware as required for your needs. KitProg2 does not work with the ModusToolbox software. Likewise, if you update to KitProg3, PSoC Creator won't work with Cypress kits until you restore KitProg2.

Note On a Linux machine, you must run the `udev_rules\install_rules.sh` script before the first run of the fw-loader.

For more details, refer to the *KitProg3 User Guide*.

Bluetooth Programming/Debugging

WICED Bluetooth applications have different launch configurations than PSoC 6 MCU applications. Bluetooth applications use External Tools configurations for programming the device, and Attach configurations for attaching to the debugger. The Program launch configurations are available from the Quick Panel when you select <app name> project. Each configuration is preceded by the application name.



Program Configuration

This launch configuration runs a script and programs the device's memories.

- **<app-name> Program:** Program is used to build and program embedded applications onto the target device.

Attach Configurations

The Attach configurations are used to attach to the debugger without first programming the device. They include:

- **<app-name> Attach_JLink**
- **<app-name> Attach_KitProg3**

Debug Settings

WICED Bluetooth devices use the JTAG SWD interface on a kit for debug via OpenOCD or J-Link probe. Typically, GPIOs need to be reprogrammed (via firmware) to enable SWD, so debug can't be performed directly after a device reset. The debugger is usually attached to a running device and symbols only are loaded.

The CYW920819EVB-02 and CYBT-213043-MESH kits have additional requirements in order to launch the ModusToolbox IDE debugger. In general, most debugging for these kits is done with logs and sniffers, since real time execution is usually needed for the protocols to run properly.

Refer to the *WICED Hardware Debugging* document (002-20504) for more details.

Mbed OS Programming/Debugging

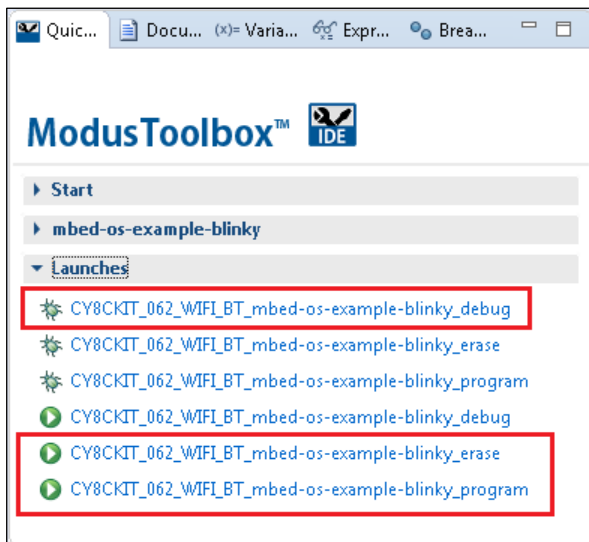
Launch Configurations

When an application is imported from Mbed OS into the ModusToolbox IDE, the tool generates the following launch configurations. There are three debug configurations and three program configurations:

`<target>_<app-name>_debug` – Programs the device as needed and launches the debugger.

`<target>_<app-name>_erase` – Erases the device.

`<target>_<app-name>_program` – Programs the device.



Note The three debug launches will attempt to switch to the debug perspective, while the three program launches will not. Cypress recommends using the debug launch for "debug" and the program launches for "erase" and "program."

Change PyOCD Message Color

Messages in the console for pyOCD display in red because they are written to stderr. The Eclipse preference for stderr defaults to red. If you want to change the color of stderr:

1. Open the Preferences dialog and select **Run/Debug > Console**.
2. Click on the option for **Standard Error text color**, select the desired color, and click **OK**.
3. Then, click **Apply and Close** to close the Preferences dialog.

Enable Peripherals View

For applications imported from Mbed OS, the **Peripherals** view does not contain any peripherals because you must provide the path to the device hardware description .SVD file. To do this:

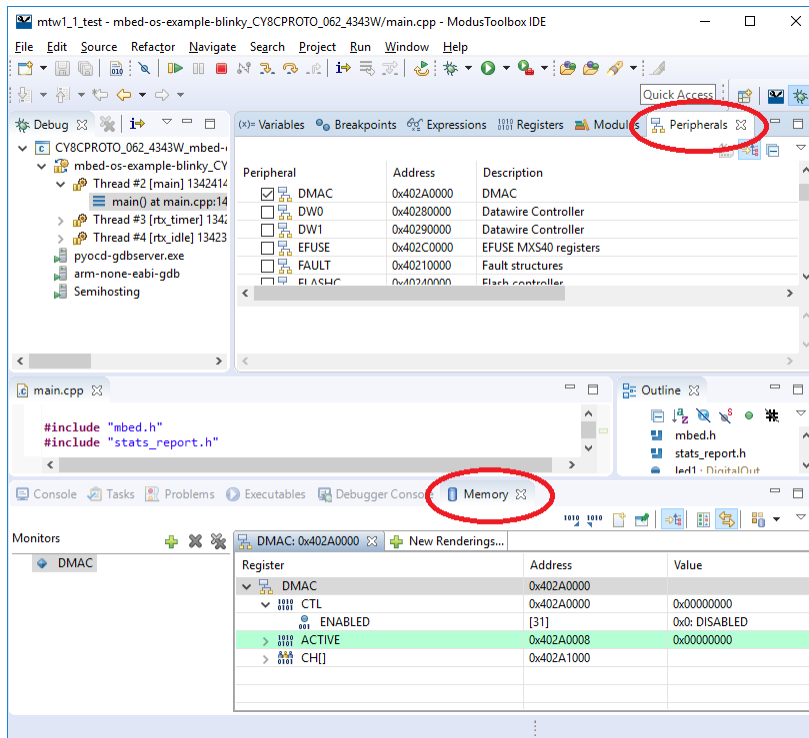
1. Open the Debug Configurations dialog and select the appropriate pyOCD debug launch configuration.

2. Select the **SVD Path** tab and provide path to the .SVD file. The following .SVD files included with ModusToolbox are available under the following paths:

PSoc6ABLE2: <install_dir>/libraries/udd-1.1/udd/devices/MXS40/PSoc6ABLE2/studio/svd/psoc6_01.svd

PSoc6A2M : Install_dir>/libraries/udd-1.1/udd/devices/MXS40/PSoc6A2M/studio/svd/psoc6_02.svd

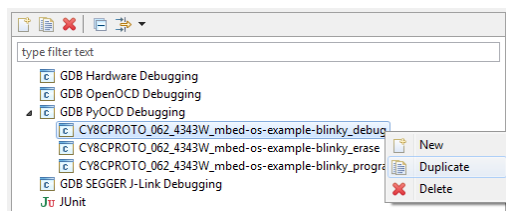
3. Click **Apply** and then click **Debug**. When the debugger halts, notice that the **Peripherals** view contains various peripherals to select, depending on the application.



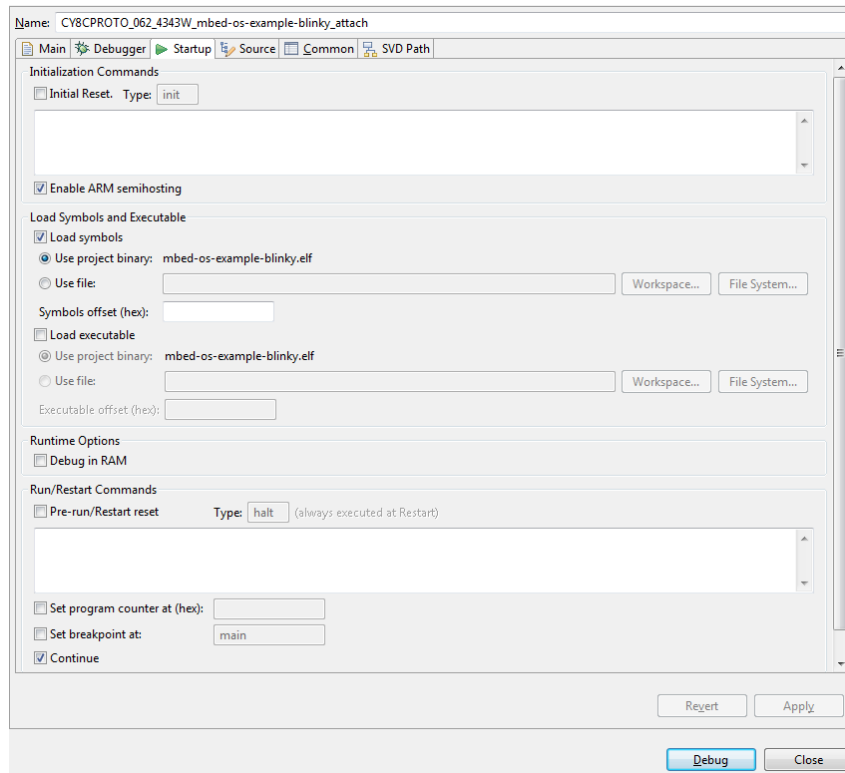
Attach to Running Target (PyOCD)

There are some cases where you may wish to attach to a running PSoC 6 target as part of a debug session. This is accomplished by copying and modifying existing debug configurations.

1. See [Import Mbed OS Applications](#) for instructions to import an example Mbed project into the ModusToolbox IDE workspace.
2. Open the Debug Configurations dialog, and duplicate the Debug configuration; for example, "CY8CPROTO_062_4343W_mbed-os-example-blinky_debug."



3. Rename the new Debug configuration to something meaningful, such as "CY8CPROTO_062_4343W_mbed-os-example-blinky_attach."

4. Go to the **Startup** tab.


- a. Unselect **Initial Reset**.
 - b. Unselect **Load executable**.
 - c. Unselect **Pre-run/Restart reset**.
 - d. Unselect **Set breakpoint at**.
 - e. Select **Continue**.
5. Ensure your KitProg3 is in DAPLink mode and your target is running. See [KitProg Firmware Loader](#).
 6. Click **Debug** to attach to the running target using the new launch configuration.