Eclipse IDE for ModusToolbox®

User Guide

Document Number: 002-24375 Rev *G

Cypress Semiconductor
An Infineon Technologies Company
198 Champion Court
San Jose, CA 95134-1709
www.cypress.com
www.infineon.com

Contents

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) that supports application configuration and development.

The ModusToolbox ecosystem includes IoT development platforms like Mbed™ OS and Amazon FreeRTOS™, in addition to the software installed on your computer and available on GitHub. For a more comprehensive description of the ModusToolbox software, refer to the *ModusToolbox User Guide*.

This document provides information about creating applications as well as building, programming, and debugging them. This guide primarily covers the Eclipse 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.

- ModusToolbox Installation Guide
- ModusToolbox User Guide
- 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: `C:\ …cd\icc\, user entered text`
Italics	Displays file names and reference documentation: *sourcefile.hex*
[**bracketed, bold**]	Displays keyboard commands in procedures: [**Enter**] or [**Ctrl**] [**C**]
File > New Application	Represents menu paths: **File > New Application > Clone**

Convention	Usage
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: Eclipse IDE for ModusToolbox 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	10/15/19	Updates for ModusToolbox version 2.0.
*C	10/29/19	Edits to various screen captures and related text.
*D	3/20/20	Updates for ModusToolbox version 2.1.
*E	4/1/20	Fix broken links.
*F	8/25/20	Updates for ModusToolbox version 2.2.
*G	10/30/20	Updated Import information. Added information about exporting/sharing an application. Added information about PSoC 4 support. Added information about restoring shared directory.

2. Getting Started

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

Launch Eclipse IDE

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

<install_path>\ModusToolbox\ide_<version>\eclipse\

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.

To launch the Eclipse IDE:

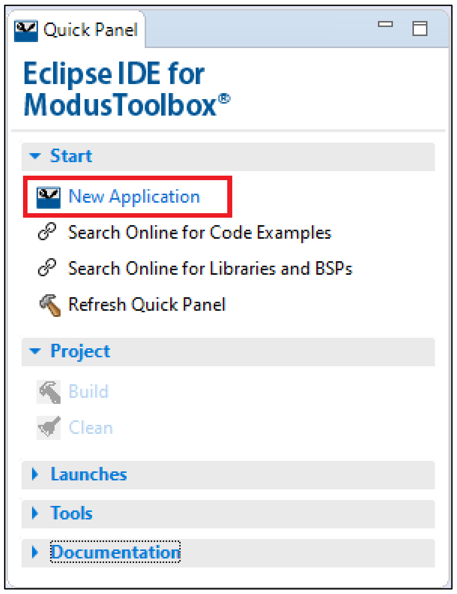
- On Windows, select the Eclipse IDE for ModusToolbox <version> item from the **Start** menu.
- For other operating systems, run the “ModusToolbox” executable file.

When launching the Eclipse 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.

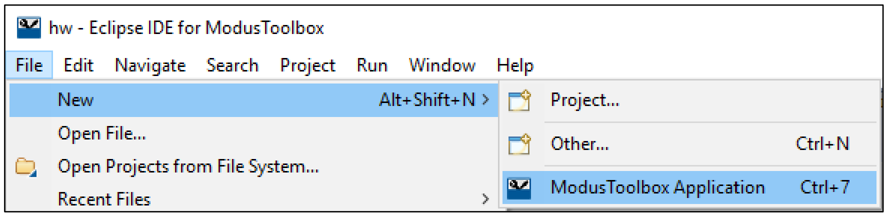
For more details about Eclipse, refer to the Eclipse documentation, as well as the Cypress Eclipse Survival Guide.

Open Project Creator Tool

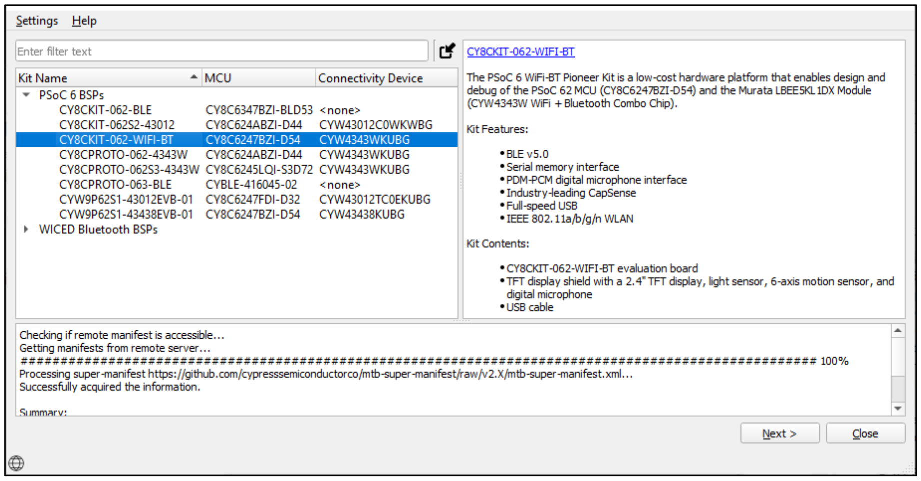
Click the **New Application** link in the Eclipse IDE Quick Panel.


Quick Panel
Eclipse IDE for ModusToolbox®
Start
New Application
Search Online for Code Examples
Search Online for Libraries and BSPs
Refresh Quick Panel
Project
Build
Clean
Launches
Tools
Documentation

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


hw - Eclipse IDE for ModusToolbox
File Edit Navigate Search Project Run Window Help
New Alt+Shift+N >
Open File...
Open Projects from File System...
Recent Files >
Project...
Other... Ctrl+N
ModusToolbox Application Ctrl+7

These commands launch the Project Creator tool, which provides several applications for use with different development kits. The kits available may change over time.



Kit Name	MCU	Connectivity Device
PSoC 6 BSPs		
CY8CKIT-062-BLE	CY8C6347BZI-BLD53	<none>
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	<none>
CYW9P62S1-43012EVB-01	CY8C6247FDI-D32	CYW43012TC0EKUBG
CYW9P62S1-43438EVB-01	CY8C6247BZI-D54	CYW43438KUBG
WICED Bluetooth BSPs		

CY8CKIT-062-WIFI-BT

The PSoC 6 WiFi-BT Pioneer Kit is a low-cost hardware platform that enables design and debug of the PSoC 62 MCU (CY8C6247BZI-D54) and the Murata LBEE5KL1DX Module (CYW4343W WiFi + Bluetooth Combo Chip).

Kit Features:

- BLE v5.0
- Serial memory interface
- PDM-PCM digital microphone interface
- Industry-leading CapSense
- Full-speed USB
- IEEE 802.11a/b/g/n WLAN

Kit Contents:

- CY8CKIT-062-WIFI-BT evaluation board
- TFT display shield with a 2.4" TFT display, light sensor, 6-axis motion sensor, and digital microphone
- USB cable

```
Checking if remote manifest is accessible...
Getting manifests from remote server...
################################################################################################# 100%
Processing super-manifest https://github.com/cypresssemiconductorco/mtb-super-manifest/raw/v2.X/mtb-super-manifest.xml...
Successfully acquired the information.

Summary:
```

For more details about using this tool, refer to the *Project Creator Guide*.

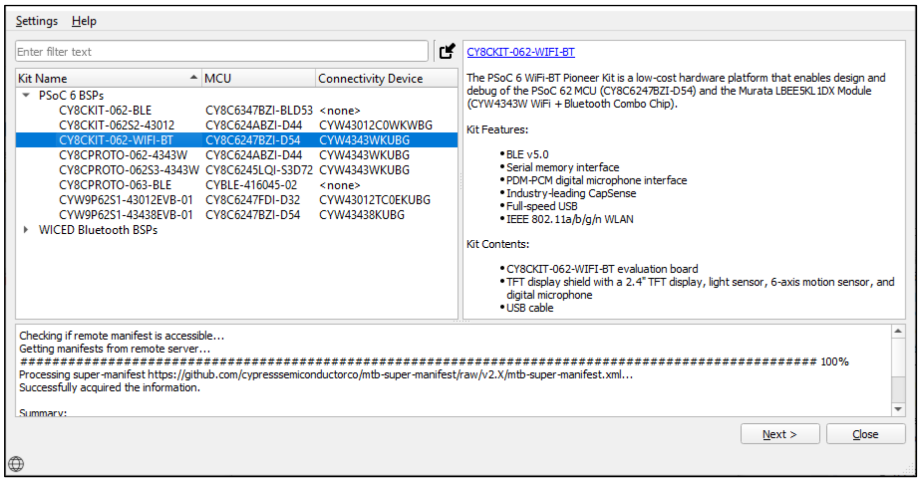
Create an Application

This section provides a walkthrough for creating a ModusToolbox application.

Choose Board Support Package

The Project Creator tool 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. Depending on the settings for your system, you may see different categories of of BSPS, inclusing PSoC 4, PSoC 6, and WICED Bluetooth.

For this example, select the CY8CKIT-062-WIFI-BT kit.



Kit Name	MCU	Connectivity Device
PSoC 6 BSPs		
CY8CKIT-062-BLE	CY8C6347BZI-BLD53	<none>
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	<none>
CYW9P62S1-43012EVB-01	CY8C6247FDI-D32	CYW43012TC0EKUBG
CYW9P62S1-43438EVB-01	CY8C6247BZI-D54	CYW43438KUBG
WICED Bluetooth BSPs		

CY8CKIT-062-WIFI-BT

The PSoC 6 WiFi-BT Pioneer Kit is a low-cost hardware platform that enables design and debug of the PSoC 62 MCU (CY8C6247BZI-D54) and the Murata LBEE5KL1DX Module (CYW4343W WiFi + Bluetooth Combo Chip).

Kit Features:

- BLE v5.0
- Serial memory interface
- PDM-PCM digital microphone interface
- Industry-leading CapSense
- Full-speed USB
- IEEE 802.11a/b/g/n WLAN

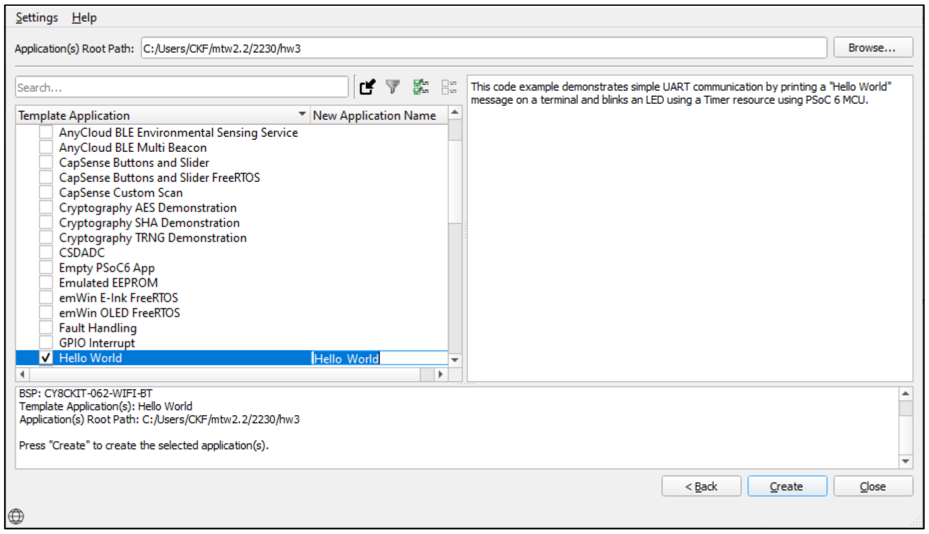
Kit Contents:

- CY8CKIT-062-WIFI-BT evaluation board
- TFT display shield with a 2.4" TFT display, light sensor, 6-axis motion sensor, and digital microphone
- USB cable

```
Checking if remote manifest is accessible...
Getting manifests from remote server...
#################################################################################################### 100%
Processing super-manifest https://github.com/cypresssemiconductorco/mtb-super-manifest/raw/v2.X/mtb-super-manifest.xml...
Successfully acquired the information.

Summary:
```

Select Application

Click **Next >** to open the Select Application page. This page lists various applications available for the selected kit. As you select an application, a description displays on the right. You can select multiple applications for the selected BSP by enabling the check box next to the applicable applications.



Application(s) Root Path: C:/Users/CKF/mtw2.2/2230/hw3

Template Application	New Application Name
☐ AnyCloud BLE Environmental Sensing Service	
☐ AnyCloud BLE Multi Beacon	
☐ CapSense Buttons and Slider	
☐ CapSense Buttons and Slider FreeRTOS	
☐ CapSense Custom Scan	
☐ Cryptography AES Demonstration	
☐ Cryptography SHA Demonstration	
☐ Cryptography TRNG Demonstration	
☐ CSDADC	
☐ Empty PSoC6 App	
☐ Emulated EEPROM	
☐ emWin E-Ink FreeRTOS	
☐ emWin OLED FreeRTOS	
☐ Fault Handling	
☐ GPIO Interrupt	
✓ Hello World	Hello World

This code example demonstrates simple UART communication by printing a "Hello World" message on a terminal and blinks an LED using a Timer resource using PSoC 6 MCU.

```
BSP: CY8CKIT-062-WIFI-BT
Template Application(s): Hello World
Application(s) Root Path: C:/Users/CKF/mtw2.2/2230/hw3

Press "Create" to create the selected application(s).
```

Note For BTSDK v2.7 and earlier, you must also create the project called “wiced_btsdk,” which contains the SDK, BSPs, and libraries that are shared among all Bluetooth applications in a workspace. Refer to the ModusToolbox 2.1 version of this document for details.

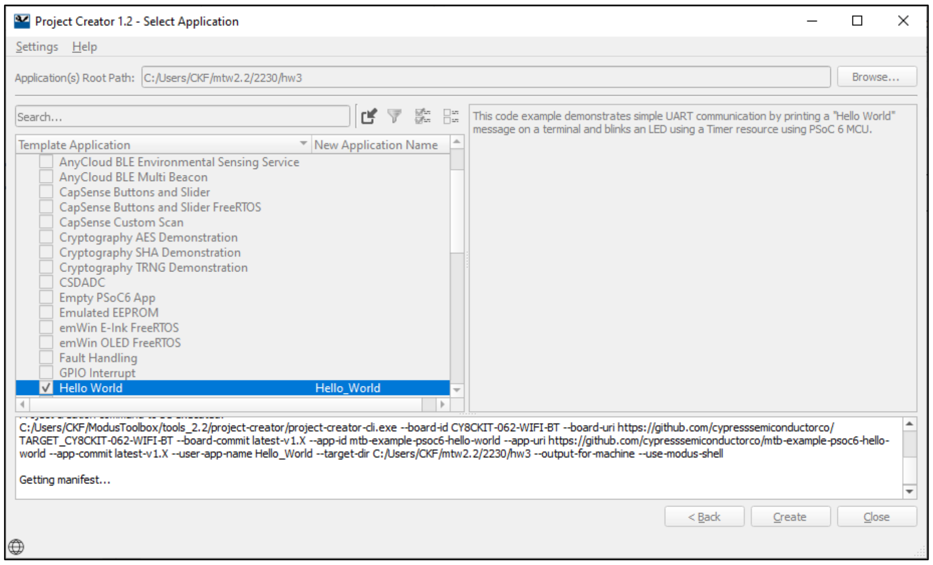
For this example:

- Select the check box next to the “Hello World” application.
- If desired, type a name for the application under **New Application Name**. Do not use spaces in the application name. In this case, we use the default “Hello_World” as the 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 dialog with all the other applications. See also Import Application/Code Example for additional details.

Create Application

Click **Create** to begin the project creation process.


Project Creator 1.2 - Select Application

Application(s) Root Path: C:/Users/CKF/mtw2.2/2230/hw3

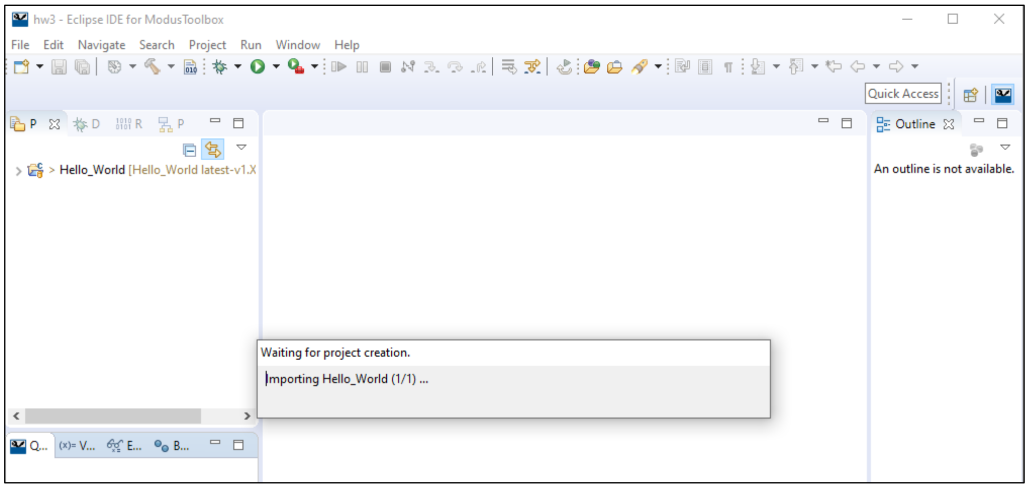
This code example demonstrates simple UART communication by printing a "Hello World" message on a terminal and blinks an LED using a Timer resource using PSoC 6 MCU.

Template Application	New Application Name
☐ AnyCloud BLE Environmental Sensing Service	
☐ AnyCloud BLE Multi Beacon	
☐ CapSense Buttons and Slider	
☐ CapSense Buttons and Slider FreeRTOS	
☐ CapSense Custom Scan	
☐ Cryptography AES Demonstration	
☐ Cryptography SHA Demonstration	
☐ Cryptography TRNG Demonstration	
☐ CSDADC	
☐ Empty PSoC6 App	
☐ Emulated EEPROM	
☐ emWin E-Ink FreeRTOS	
☐ emWin OLED FreeRTOS	
☐ Fault Handling	
☐ GPIO Interrupt	
✓ Hello World	Hello_World

```
C:/Users/CKF/ModusToolbox/tools_2.2/project-creator/project-creator-cli.exe --board-id CY8CKIT-062-WIFI-BT --board-uri https://github.com/cypresssemiconductorco/
TARGET_CY8CKIT-062-WIFI-BT --board-commit latest-v1.X --app-id mtb-example-psoc6-hello-world --app-uri https://github.com/cypresssemiconductorco/mtb-example-psoc6-hello-
world --app-commit latest-v1.X --user-app-name Hello_World --target-dir C:/Users/CKF/mtw2.2/2230/hw3 --output-for-machine --use-modus-shell

Getting manifest...
```

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 Project Creator tool closes automatically. In the IDE, a message displays about importing the project:

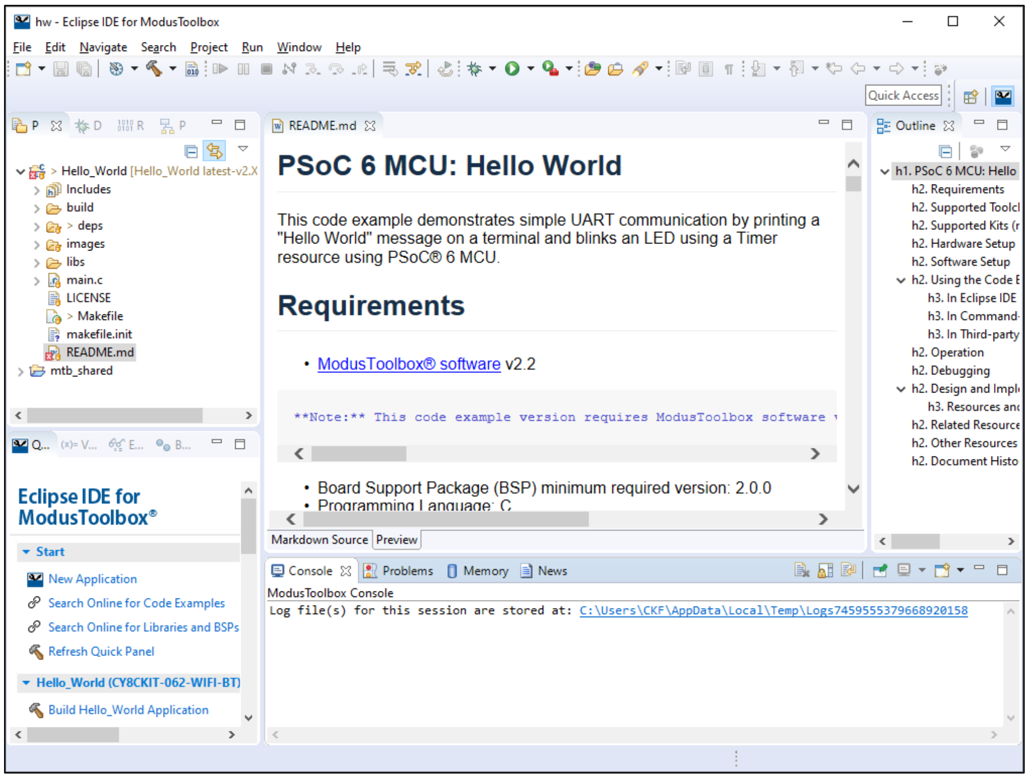

hw3 - Eclipse IDE for ModusToolbox

Hello_World [Hello_World latest-v1.X

An outline is not available.

```
Waiting for project creation.
Importing Hello_World (1/1) ...
```

After several moments, the application opens with the *Hello_World* in the Project Explorer, and the *README.md* file opens in the file viewer.



PSoC 6 MCU: Hello World

This code example demonstrates simple UART communication by printing a "Hello World" message on a terminal and blinks an LED using a Timer resource using PSoC® 6 MCU.

Requirements

- ModusToolbox® software v2.2

```
**Note:** This code example version requires ModusToolbox software v
```

- Board Support Package (BSP) minimum required version: 2.0.0
- Programming Language: C

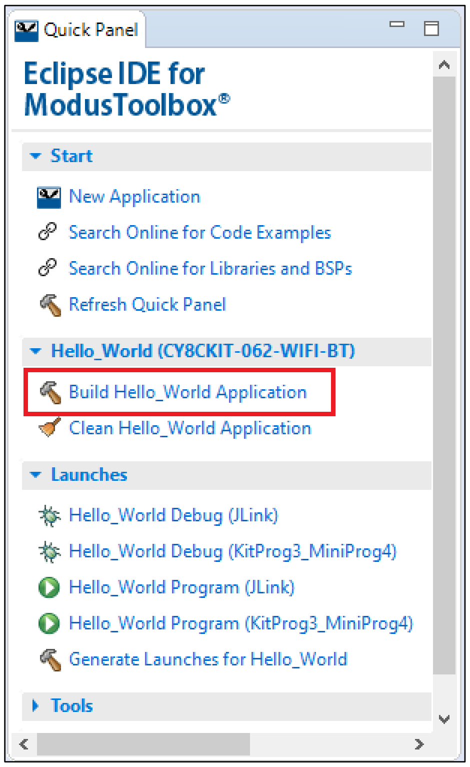
```
ModusToolbox Console
Log file(s) for this session are stored at: C:\Users\CKF\AppData\Local\Temp\Logs7459555379668920158
```

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

Build Application

After loading an application, build it to generate the necessary files. Select a project. Then, in the **Quick Panel**, click the "Build <project> Application" link. The following images show the Quick Panel for a typical PSoC MCU application and a WICED Bluetooth application.

PSoC MCU Application



Quick Panel

Eclipse IDE for ModusToolbox®

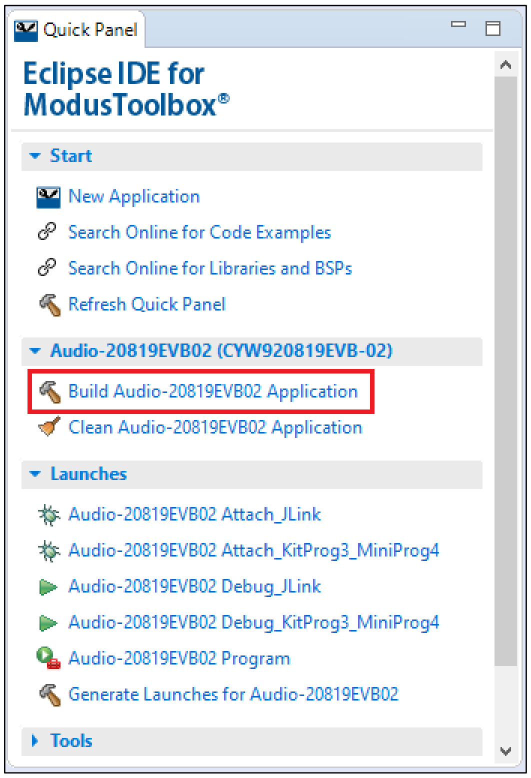
Start
- New Application
- Search Online for Code Examples
- Search Online for Libraries and BSPs
- Refresh Quick Panel

Hello_World (CY8CKIT-062-WIFI-BT)
- Build Hello_World Application
- Clean Hello_World Application

Launches
- Hello_World Debug (JLink)
- Hello_World Debug (KitProg3_MiniProg4)
- Hello_World Program (JLink)
- Hello_World Program (KitProg3_MiniProg4)
- Generate Launches for Hello_World

Tools

WICED Bluetooth Application



Quick Panel

Eclipse IDE for ModusToolbox®

Start
- New Application
- Search Online for Code Examples
- Search Online for Libraries and BSPs
- Refresh Quick Panel

Audio-20819EVB02 (CYW920819EVB-02)
- Build Audio-20819EVB02 Application
- Clean Audio-20819EVB02 Application

Launches
- Audio-20819EVB02 Attach_JLink
- Audio-20819EVB02 Attach_KitProg3_MiniProg4
- Audio-20819EVB02 Debug_JLink
- Audio-20819EVB02 Debug_KitProg3_MiniProg4
- Audio-20819EVB02 Program
- Generate Launches for Audio-20819EVB02

Tools

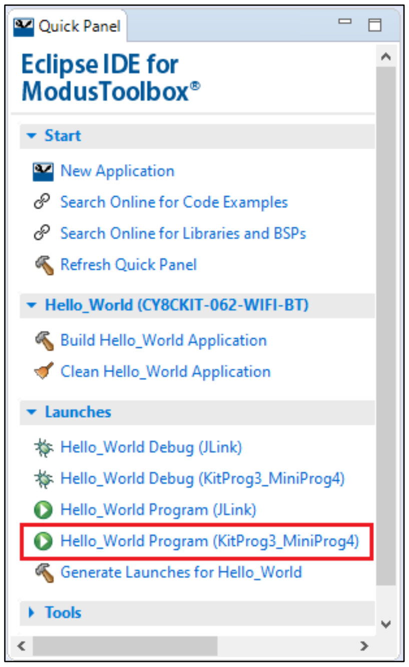
Messages display in the Console, indicating whether the build was successful or not. For more details about building applications and the various Consoles available, see the Build Applications chapter.

Program 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_MiniProg4)" link for a PSoC MCU application, and "<app-name> Program" for a WICED Bluetooth application.

PSoC MCU Application

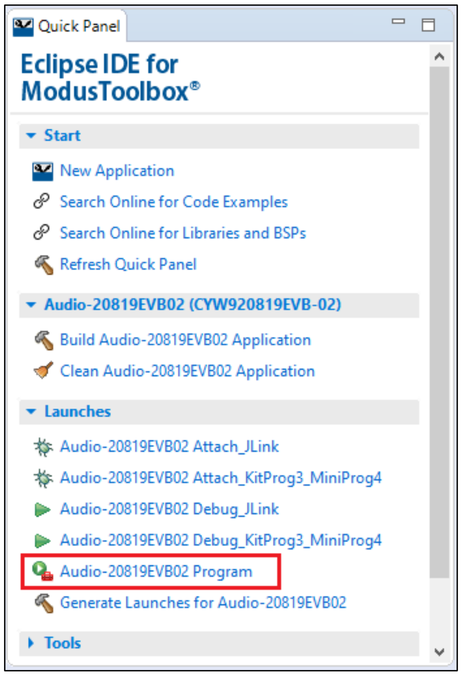


Quick Panel

Eclipse IDE for ModusToolbox®

- Start
  - New Application
  - Search Online for Code Examples
  - Search Online for Libraries and BSPs
  - Refresh Quick Panel
- Hello_World (CY8CKIT-062-WIFI-BT)
  - Build Hello_World Application
  - Clean Hello_World Application
- Launches
  - Hello_World Debug (JLink)
  - Hello_World Debug (KitProg3_MiniProg4)
  - Hello_World Program (JLink)
  - Hello_World Program (KitProg3_MiniProg4)
  - Generate Launches for Hello_World
- Tools

WICED Bluetooth Application



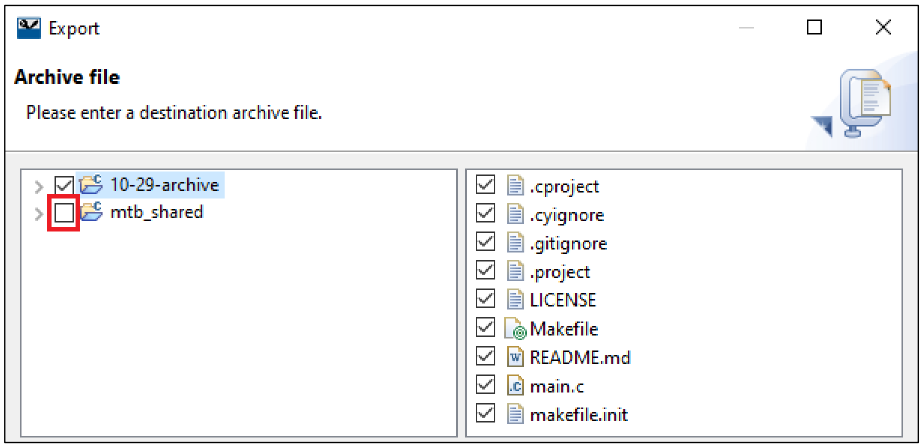
Quick Panel

Eclipse IDE for ModusToolbox®

- Start
  - New Application
  - Search Online for Code Examples
  - Search Online for Libraries and BSPs
  - Refresh Quick Panel
- Audio-20819EVB02 (CYW920819EVB-02)
  - Build Audio-20819EVB02 Application
  - Clean Audio-20819EVB02 Application
- Launches
  - Audio-20819EVB02 Attach_JLink
  - Audio-20819EVB02 Attach_KitProg3_MiniProg4
  - Audio-20819EVB02 Debug_JLink
  - Audio-20819EVB02 Debug_KitProg3_MiniProg4
  - Audio-20819EVB02 Program
  - Generate Launches for Audio-20819EVB02
- Tools

Export/Share Application

The Eclipse IDE has several methods to export and share applications, and ModusToolbox supports all of them. Keep in mind that a typical ModusToolbox application includes many libraries that are available on GitHub. Those libraries are updated during the `make getlibs` command that the Project Creator tool and Library Manager tool run as part of their operations. Therefore, you can substantially reduce the size of your exported application by excluding the libraries. They can be regenerated by the recipient of the imported application. The following image shows the Eclipse IDE Export dialog with the *mtb_shared* directory de-selected in order to exclude the libraries.



Export

Archive file

Please enter a destination archive file.

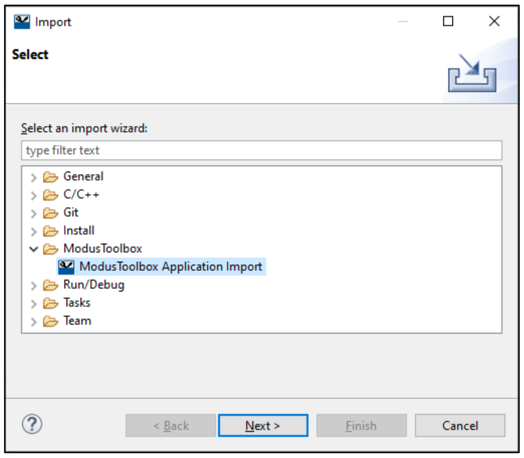
- [✓] 10-29-archive
- [ ] mtb_shared

- [✓] .cproject
- [✓] .cyignore
- [✓] .gitignore
- [✓] .project
- [✓] LICENSE
- [✓] Makefile
- [✓] README.md
- [✓] main.c
- [✓] makefile.init

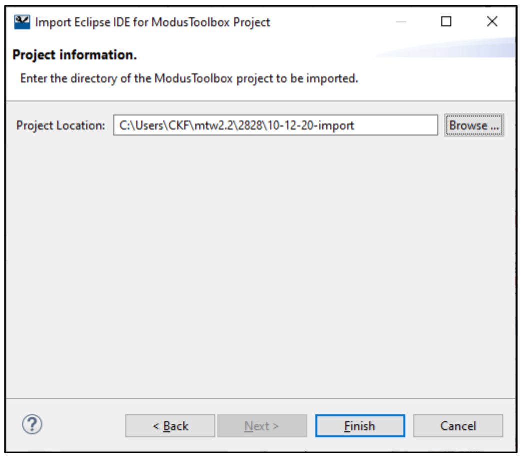
Import Application/Code Example

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 Eclipse IDE:

- **Project Creator Import Option:** Use the Project Creator tool to create a new application, and in that process select the **Import** button to select a folder that contains the application to import.
- **Eclipse IDE Import Option:** If you have an Eclipse-ready code example (for example, a project exported from Eclipse) that you want to import into the Eclipse IDE, use the **File > Import** process. Note the following:
  - On the Import dialog, select the **ModusToolbox > ModusToolbox Application Import** option.



  - On the next page, click the **Browse…** button, navigate to the application directory, and click **Select Folder**.



Enter the directory of the ModusToolbox project to be imported.

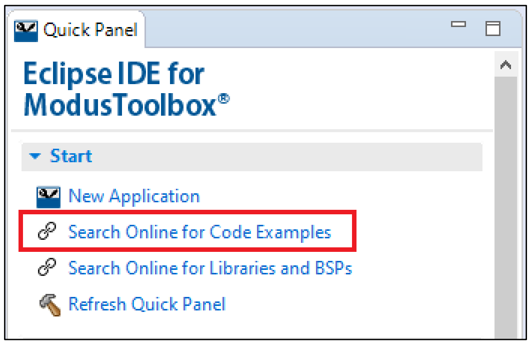
Project Location: C:\Users\CKF\mtw2.2\2828\10-12-20-import

  - Click **Finish** to begin the import process. This will take a few moments, and then the application will display in the Eclipse IDE Project Explorer.
  - If the Console displays a message, such as "Error creating Eclipse configurations," open the Library Manager and click **Update**. This runs the `make getlibs` operation to generate the necessary files and libraries.

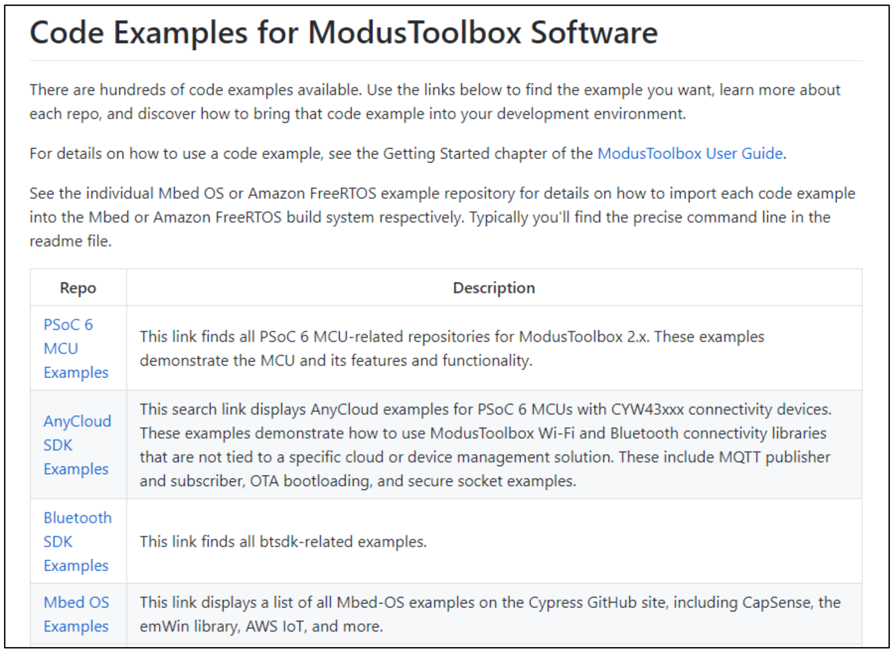
Note There are various ways to import examples into Eclipse. If you prefer a different method, make sure that all of the project files are copied into the workspace directory.

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


Quick Panel
Eclipse IDE for ModusToolbox®
Start
New Application
Search Online for Code Examples
Search Online for Libraries and BSPs
Refresh Quick Panel

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. 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.

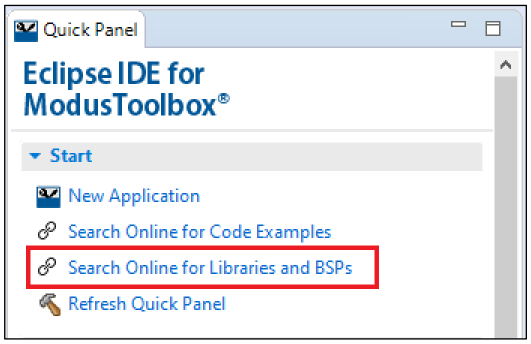
For details on how to use a code example, see the Getting Started chapter of the ModusToolbox User Guide.

See the individual Mbed OS or Amazon FreeRTOS example repository for details on how to import each code example into the Mbed or Amazon FreeRTOS build system respectively. Typically you'll find the precise command line in the readme file.

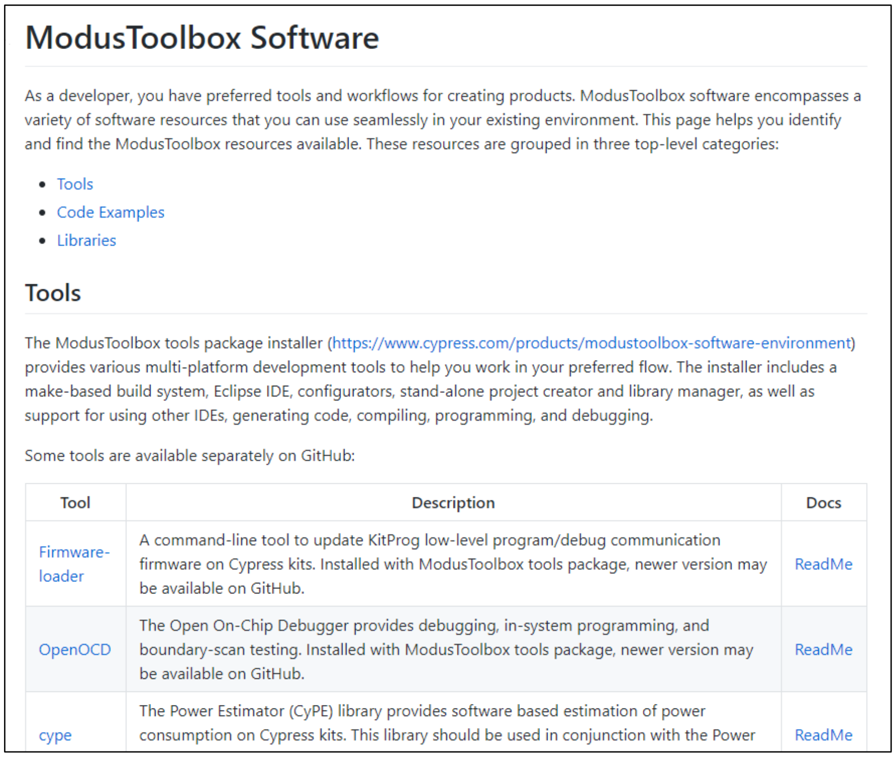
Repo	Description
PSoC 6 MCU Examples	This link finds all PSoC 6 MCU-related repositories for ModusToolbox 2.x. These examples demonstrate the MCU and its features and functionality.
AnyCloud SDK Examples	This search link displays AnyCloud examples for PSoC 6 MCUs with CYW43xxx connectivity devices. These examples demonstrate how to use ModusToolbox Wi-Fi and Bluetooth connectivity libraries that are not tied to a specific cloud or device management solution. These include MQTT publisher and subscriber, OTA bootloading, and secure socket examples.
Bluetooth SDK Examples	This link finds all btsdk-related examples.
Mbed OS Examples	This link displays a list of all Mbed-OS examples on the Cypress GitHub site, including CapSense, the emWin library, AWS IoT, and more.

Search Online for Libraries/BSPs

Cypress also provides all the libraries and BSPs online at GitHub. The **Quick Panel** provides a link to access these. Click the "Search Online for Libraries and BSPs" link.


Quick Panel
Eclipse IDE for ModusToolbox®
Start
New Application
Search Online for Code Examples
Search Online for Libraries and BSPs
Refresh Quick Panel

This opens a web browser to the GitHub repository that shows the ModusToolbox Software page.



ModusToolbox Software

As a developer, you have preferred tools and workflows for creating products. ModusToolbox software encompasses a variety of software resources that you can use seamlessly in your existing environment. This page helps you identify and find the ModusToolbox resources available. These resources are grouped in three top-level categories:

- Tools
- Code Examples
- Libraries

Tools

The ModusToolbox tools package installer (https://www.cypress.com/products/modustoolbox-software-environment) provides various multi-platform development tools to help you work in your preferred flow. The installer includes a make-based build system, Eclipse IDE, configurators, stand-alone project creator and library manager, as well as support for using other IDEs, generating code, compiling, programming, and debugging.

Some tools are available separately on GitHub:

Tool	Description	Docs
Firmware-loader	A command-line tool to update KitProg low-level program/debug communication firmware on Cypress kits. Installed with ModusToolbox tools package, newer version may be available on GitHub.	ReadMe
OpenOCD	The Open On-Chip Debugger provides debugging, in-system programming, and boundary-scan testing. Installed with ModusToolbox tools package, newer version may be available on GitHub.	ReadMe
cype	The Power Estimator (CyPE) library provides software based estimation of power consumption on Cypress kits. This library should be used in conjunction with the Power	ReadMe

3. Mbed OS to ModusToolbox Flow

For ModusToolbox 2.0 and later, 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 Eclipse 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 Eclipse IDE. The main steps include:

- Install and Configure Mbed CLI
- Switch Kit to DAPLink Mode
- Create Application in Mbed CLI
- Compile Application in Mbed CLI (Optional)
- Export Application from Mbed CLI
- Import Mbed OS Application into Eclipse IDE
- Configure Eclipse IDE
- Build, Program, Debug Application

Install and Configure Mbed CLI

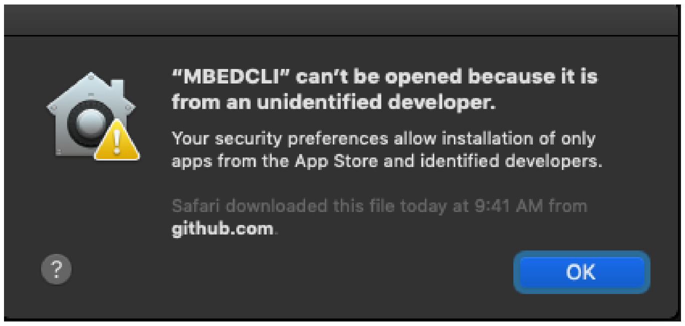
You should be familiar with Arm Mbed OS and the command line interface (CLI). Refer to the official Mbed OS instructions:

Mbed OS Website

Note: ModusToolbox version 2.2 and later only support Python version 3. As of this writing, the Mbed OS installers provide Python version 2.7. Refer to the ModusToolbox Installation Guide for details.

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.

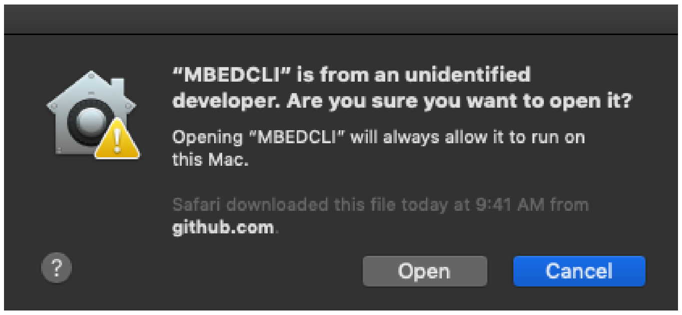
For macOS, you will get an error that it is from an unidentified developer.


“MBEDCLI” can’t be opened because it is from an unidentified developer.

Your security preferences allow installation of only apps from the App Store and identified developers.

Safari downloaded this file today at 9:41 AM from github.com.

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


"MBEDCLI" is from an unidentified developer. Are you sure you want to open it?

Opening "MBEDCLI" will always allow it to run on this Mac.

Safari downloaded this file today at 9:41 AM from github.com.

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

ModusToolbox GCC

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

- For Windows and Linux:

  ```
  mbed config -G GCC_ARM_PATH ~/ModusToolbox/tools_2.2/gcc/bin
  ```

- For macOS:

  ```
  mbed config -G GCC_ARM_PATH /Applications/ModusToolbox/tools_2.2/gcc/bin
  ```

Other Compilers

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 ARMC6_PATH "C:\Program Files\ARMCompiler6.11\bin"
mbed config -G IAR_PATH "C:\Program Files (x86)\IAR Systems\Embedded Workbench 7.5\arm"
```

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 ARMC6_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 guide is located on the https://www.cypress.com/products/psoc-programming-solutions webpage, under "Documentation." The tool is located in the following directory, by default:

<install_dir>/tools_2.2/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]    190013016c121817036c12180000000000000002e127069
  ```

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
```

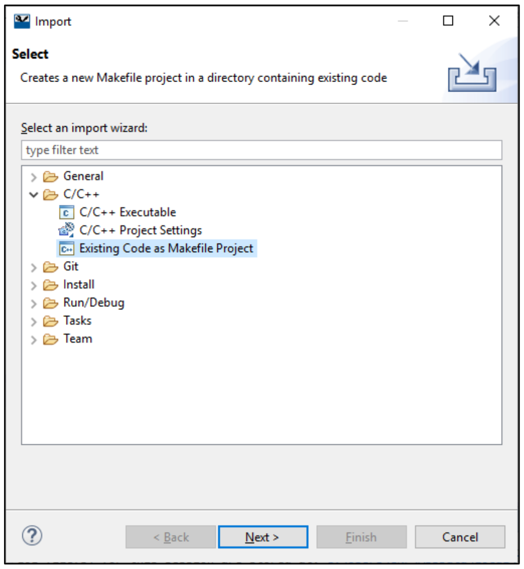
Export Application from Mbed CLI

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

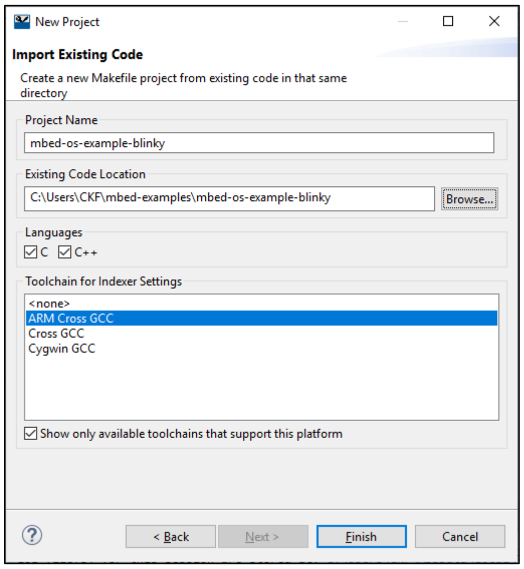
```
mbed export -i eclipse_gcc_arm -m CY8CKIT_062_WIFI_BT
```

Import Mbed OS Application into Eclipse 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**.


New Project
Import Existing Code
Create a new Makefile project from existing code in that same directory
Browse...
< Back
Next >
Finish
Cancel

Project Name: mbed-os-example-blinky

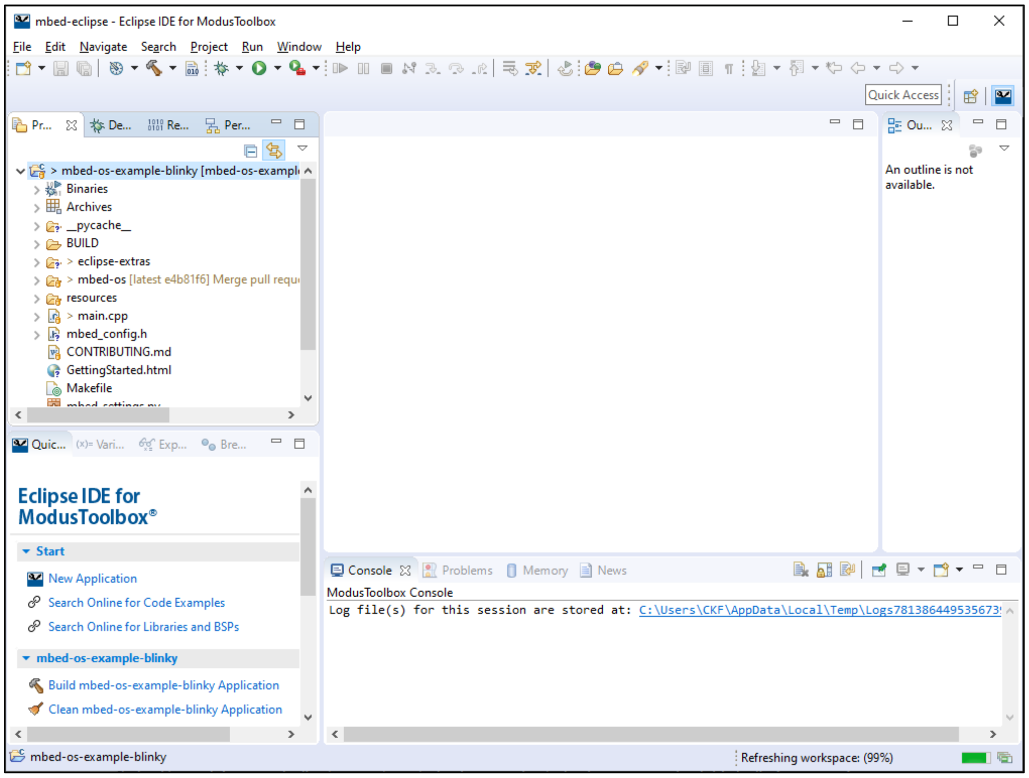
Existing Code Location: C:\Users\CKF\mbed-examples\mbed-os-example-blinky

Languages: ☑ C ☑ C++

Toolchain for Indexer Settings:
- <none>
- [✓ ARM Cross GCC]
- Cross GCC
- Cygwin GCC

☑ Show only available toolchains that support this platform

When the import finishes, the Mbed OS application will be shown in the Project Explorer.

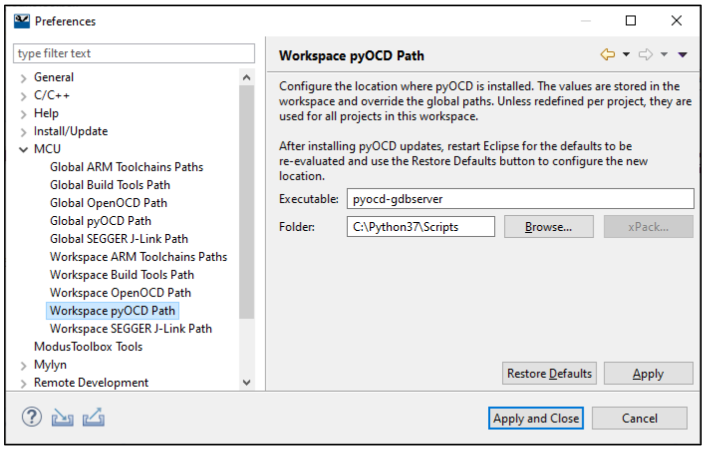

mbed-eclipse - Eclipse IDE for ModusToolbox
File
Edit
Navigate
Search
Project
Run
Window
Help
Quick Access
Pr...
De...
Re...
Per...
> mbed-os-example-blinky [mbed-os-exampl
Binaries
Archives
__pycache__
BUILD
> eclipse-extras
> mbed-os [latest e4b81f6] Merge pull requ
resources
> main.cpp
mbed_config.h
CONTRIBUTING.md
GettingStarted.html
Makefile
Ou...
An outline is not available.
Quic...
Vari...
Exp...
Bre...
Eclipse IDE for ModusToolbox®
Start
New Application
Search Online for Code Examples
Search Online for Libraries and BSPs
mbed-os-example-blinky
Build mbed-os-example-blinky Application
Clean mbed-os-example-blinky Application
Console
Problems
Memory
News
mbed-os-example-blinky
Refreshing workspace: (99%)

```
ModusToolbox Console
Log file(s) for this session are stored at: C:\Users\CKF\AppData\Local\Temp\Logs781386449535673
```

Configure Eclipse 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):



Configure the location where pyOCD is installed. The values are stored in the workspace and override the global paths. Unless redefined per project, they are used for all projects in this workspace.

After installing pyOCD updates, restart Eclipse for the defaults to be re-evaluated and use the Restore Defaults button to configure the new location.

Executable: pyocd-gdbserver

Folder: C:\Python37\Scripts

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:\Python37\Scripts

macOS:

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

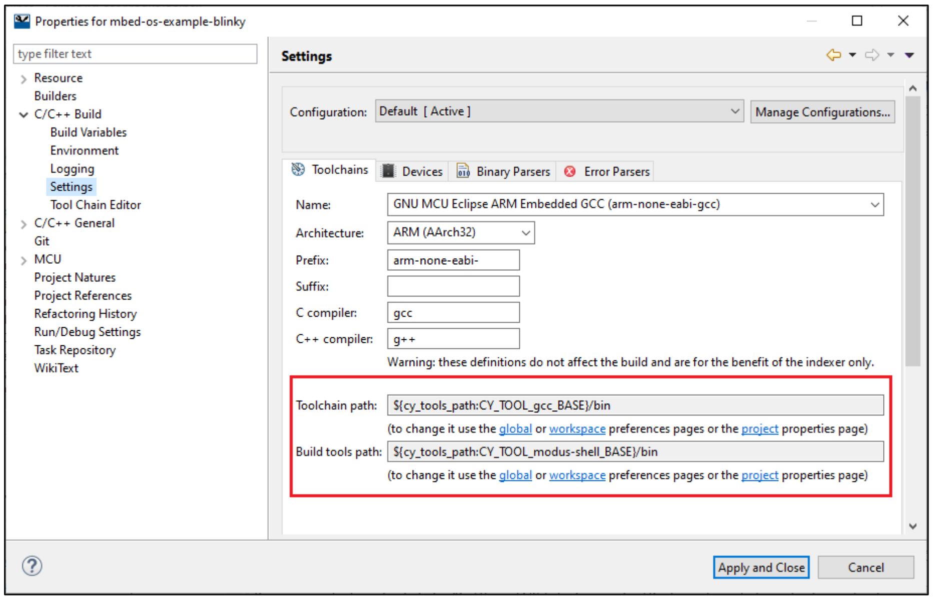
Linux:

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

After updating the settings, click **Apply and Close**.

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 vales. Click **Apply** and then click **Apply and Close**.



Configuration: Default [Active]

Name: GNU MCU Eclipse ARM Embedded GCC (arm-none-eabi-gcc)
Architecture: ARM (AArch32)
Prefix: arm-none-eabi-
Suffix:
C compiler: gcc
C++ compiler: g++
Warning: these definitions do not affect the build and are for the benefit of the indexer only.

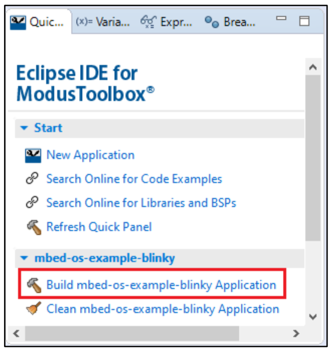
Toolchain path: ${cy_tools_path:CY_TOOL_gcc_BASE}/bin
(to change it use the global or workspace preferences pages or the project properties page)
Build tools path: ${cy_tools_path:CY_TOOL_modus-shell_BASE}/bin
(to change it use the global or workspace preferences pages or the project properties page)

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

Build, Program, Debug Application

Build

In the Quick Panel, click the “Build mbed-os-example-blinky Application” link to build the project and generate the HEX file.

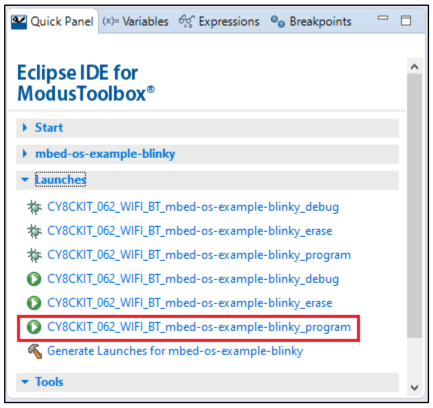

Quic...
Varia...
Expr...
Brea...
Eclipse IDE for ModusToolbox®
Start
New Application
Search Online for Code Examples
Search Online for Libraries and BSPs
Refresh Quick Panel
mbed-os-example-blinky
Build mbed-os-example-blinky Application
Clean mbed-os-example-blinky Application

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
===== bin file ready to flash: BUILD/mbed-os-example-blinky.bin =====
arm-none-eabi-objcopy -O ihex mbed-os-example-blinky.elf mbed-os-example-blinky.hex

12:03:47 Build Finished. 0 errors, 110 warnings. (took 1h:21m:30s.902ms)
```

Program

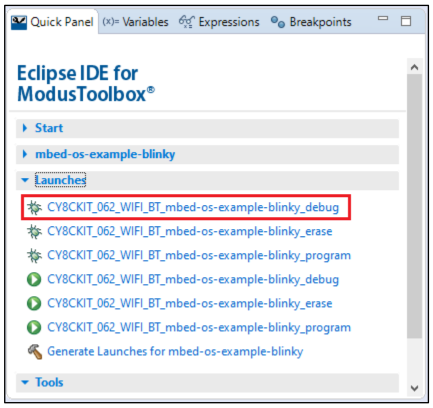
In the Quick Panel, under **Launches**, click the “CY8CKIT_062_WIFI_BT_mbed-os-example-blinky_program” link.


Quick Panel
(x)= Variables
Expressions
Breakpoints
Eclipse IDE for ModusToolbox®
Start
mbed-os-example-blinky
Launches
CY8CKIT_062_WIFI_BT_mbed-os-example-blinky_debug
CY8CKIT_062_WIFI_BT_mbed-os-example-blinky_erase
CY8CKIT_062_WIFI_BT_mbed-os-example-blinky_program
CY8CKIT_062_WIFI_BT_mbed-os-example-blinky_debug
CY8CKIT_062_WIFI_BT_mbed-os-example-blinky_erase
CY8CKIT_062_WIFI_BT_mbed-os-example-blinky_program
Generate Launches for mbed-os-example-blinky
Tools

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.

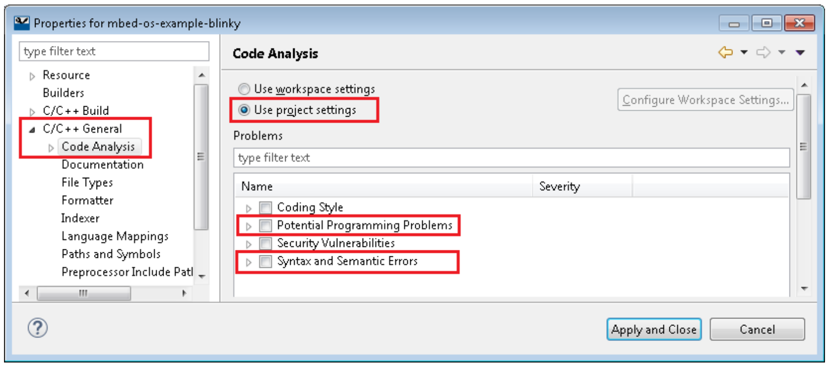
Debug

In the Quick Panel, under **Launches**, click the “CY8CKIT_062_WIFI_BT_mbed-os-example-blinky_debug” link.


Quick Panel
(x)= Variables
Expressions
Breakpoints
Eclipse IDE for ModusToolbox®
Start
mbed-os-example-blinky
Launches
CY8CKIT_062_WIFI_BT_mbed-os-example-blinky_debug
CY8CKIT_062_WIFI_BT_mbed-os-example-blinky_erase
CY8CKIT_062_WIFI_BT_mbed-os-example-blinky_program
CY8CKIT_062_WIFI_BT_mbed-os-example-blinky_debug
CY8CKIT_062_WIFI_BT_mbed-os-example-blinky_erase
CY8CKIT_062_WIFI_BT_mbed-os-example-blinky_program
Generate Launches for mbed-os-example-blinky
Tools

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."


Properties for mbed-os-example-blinky

type filter text
- Resource
- Builders
- C/C++ Build
- C/C++ General
  - Code Analysis
  - Documentation
  - File Types
  - Formatter
  - Indexer
  - Language Mappings
  - Paths and Symbols
  - Preprocessor Include Pat

Code Analysis

( ) Use workspace settings
(●) Use project settings

Configure Workspace Settings...

Problems

type filter text

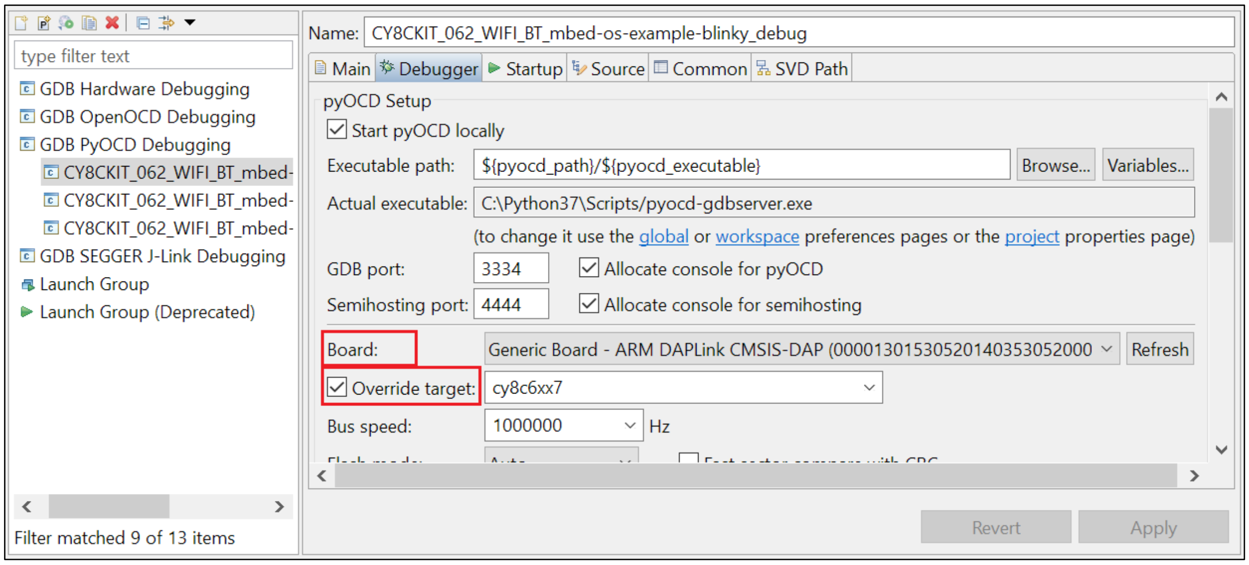
Name	Severity
☐ Coding Style	
☐ Potential Programming Problems	
☐ Security Vulnerabilities	
☐ Syntax and Semantic Errors	

Apply and Close | Cancel

Using MiniProg4 in DAPLink Mode

You can use MiniProg4 as a probe in DAPLink mode for programming/debugging an Mbed OS project imported into the Eclipse IDE. You switch the MiniProg4 to DAPLink mode in the same way as for KitProg3. See Switch Kit to DAPLink Mode.

After importing an Mbed OS Application into the Eclipse IDE, make additional changes to the appropriate launch configurations. Under the **Debugger** tab:


type filter text
- GDB Hardware Debugging
- GDB OpenOCD Debugging
- GDB PyOCD Debugging
  - CY8CKIT_062_WIFI_BT_mbed-
  - CY8CKIT_062_WIFI_BT_mbed-
  - CY8CKIT_062_WIFI_BT_mbed-
- GDB SEGGER J-Link Debugging
- Launch Group
- Launch Group (Deprecated)

Filter matched 9 of 13 items

Name: CY8CKIT_062_WIFI_BT_mbed-os-example-blinky_debug

Main | Debugger | Startup | Source | Common | SVD Path

pyOCD Setup

☑ Start pyOCD locally

Executable path: ${pyocd_path}/${pyocd_executable} Browse... Variables...

Actual executable: C:\Python37\Scripts\pyocd-gdbserver.exe

(to change it use the global or workspace preferences pages or the project properties page)

GDB port: 3334 ☑ Allocate console for pyOCD

Semihosting port: 4444 ☑ Allocate console for semihosting

Board: Generic Board - ARM DAPLink CMSIS-DAP (00001301530520140353052000 Refresh

☑ Override target: cy8c6xx7

Bus speed: 1000000 Hz

Revert | Apply

- Select MiniProg4 as the **Board**. In the drop-down list it will be shown as "Generic Board - ARM DAPLink CMSIS-DAP (00001301..."
- Select the **Override target** check box.
- Select/enter the appropriate target name to the right of **Override target**. You can run the following command from a terminal window to get a list of supported targets:

  ```
  pyocd list -t -r Cypress
  ```

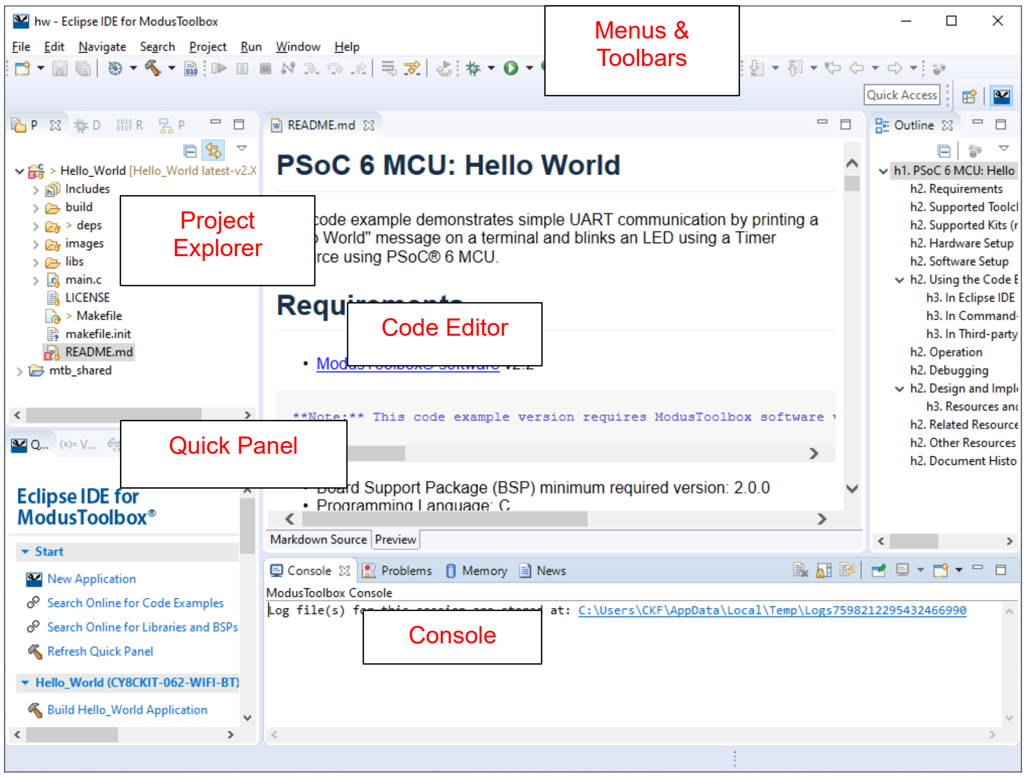
- Click **Apply** to apply changes.

4. IDE Description

Overview

The 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 IDE Survival Guide*, which provides tips and hints for how to use the Eclipse 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 Eclipse IDE also provides a Welcome Page, which displays on first launch of the IDE for a given workspace.

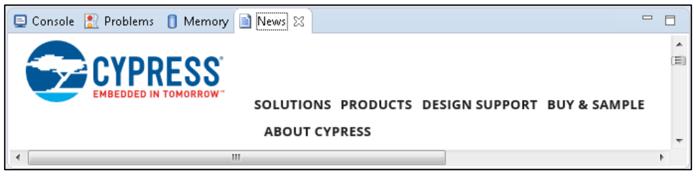


```
Log file(s) for this session are stored at: C:\Users\CKF\AppData\Local\Temp\Logs7598212295432466990
```

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.

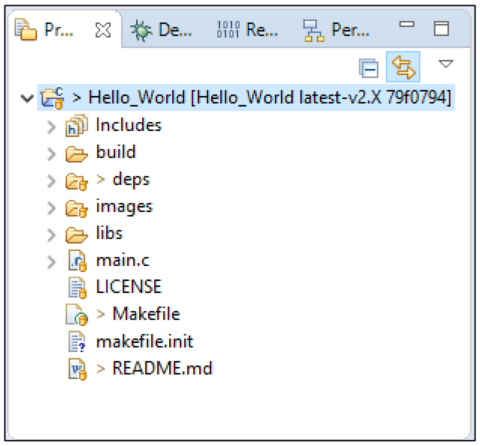


- Code Editor – Use the Code Editor to edit various source files in your application.

Project Explorer

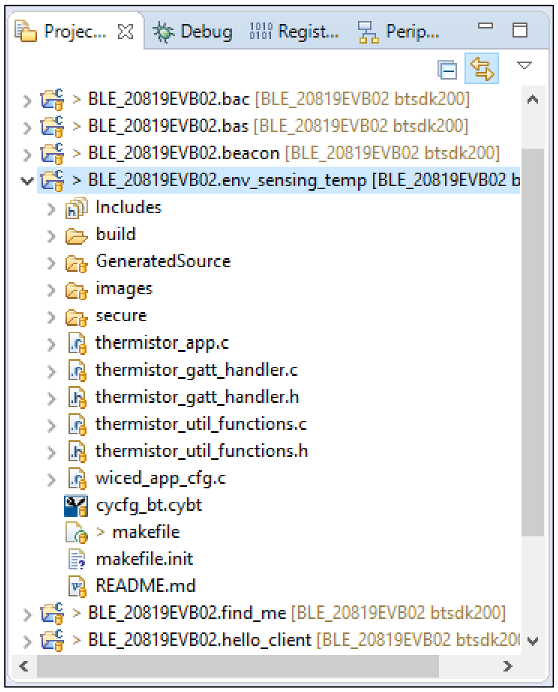
In the Eclipse IDE, after creating an application, the Project Explorer contains one or more related project folders. The following images show a PSoC MCU application and a WICED Bluetooth application. For applications imported from Mbed OS, see Mbed OS to ModusToolbox Flow.

PSoC MCU Application



```
> Hello_World [Hello_World latest-v2.X 79f0794]
    Includes
    build
    > deps
    images
    libs
    main.c
    LICENSE
    > Makefile
    makefile.init
    > README.md
```

WICED Bluetooth Application



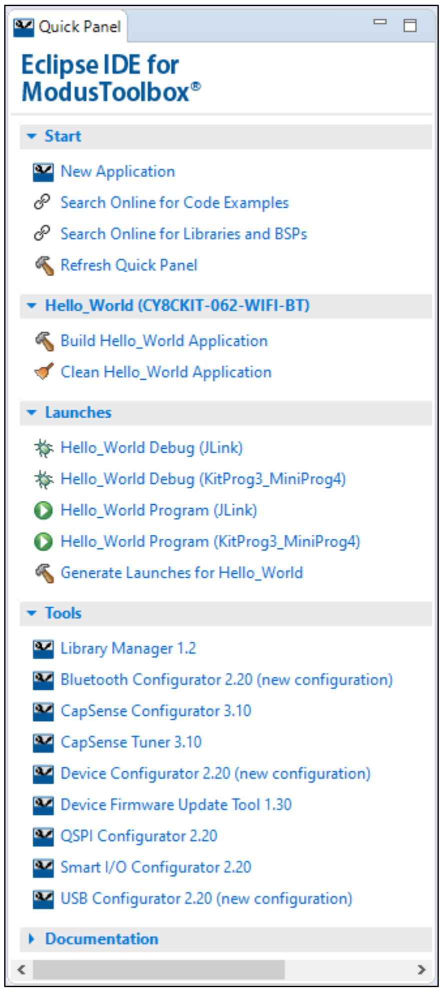
```
> BLE_20819EVB02.bac [BLE_20819EVB02 btsdk200]
> BLE_20819EVB02.bas [BLE_20819EVB02 btsdk200]
> BLE_20819EVB02.beacon [BLE_20819EVB02 btsdk200]
> BLE_20819EVB02.env_sensing_temp [BLE_20819EVB02 b
    Includes
    build
    GeneratedSource
    images
    secure
    thermistor_app.c
    thermistor_gatt_handler.c
    thermistor_gatt_handler.h
    thermistor_util_functions.c
    thermistor_util_functions.h
    wiced_app_cfg.c
    cycfg_bt.cybt
    > makefile
    makefile.init
    README.md
> BLE_20819EVB02.find_me [BLE_20819EVB02 btsdk200]
> BLE_20819EVB02.hello_client [BLE_20819EVB02 btsdk20
```

Both types of applications contain a similar project structure. Each contains the main application source code, and a Makefile. Note that PSoC MCU applications contain a libs directory, while 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 MCU Application



Quick Panel

Eclipse IDE for ModusToolbox®

Start
- New Application
- Search Online for Code Examples
- Search Online for Libraries and BSPs
- Refresh Quick Panel

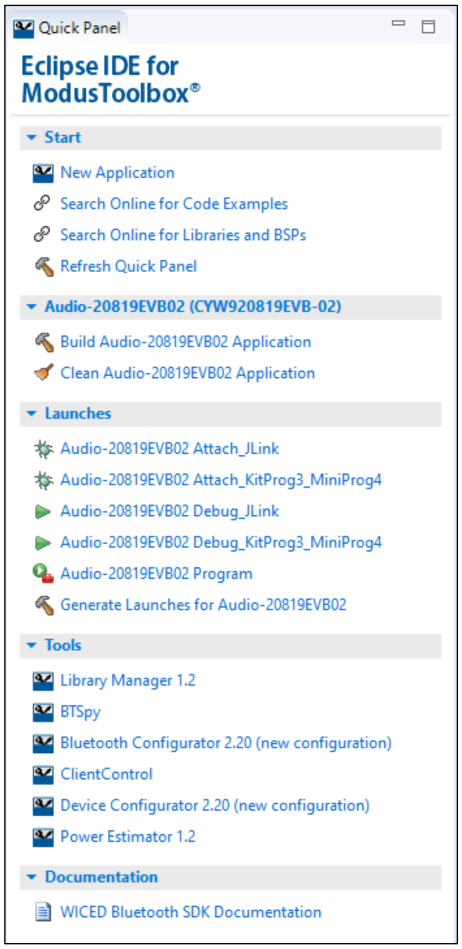
Hello_World (CY8CKIT-062-WIFI-BT)
- Build Hello_World Application
- Clean Hello_World Application

Launches
- Hello_World Debug (JLink)
- Hello_World Debug (KitProg3_MiniProg4)
- Hello_World Program (JLink)
- Hello_World Program (KitProg3_MiniProg4)
- Generate Launches for Hello_World

Tools
- Library Manager 1.2
- Bluetooth Configurator 2.20 (new configuration)
- CapSense Configurator 3.10
- CapSense Tuner 3.10
- Device Configurator 2.20 (new configuration)
- Device Firmware Update Tool 1.30
- QSPI Configurator 2.20
- Smart I/O Configurator 2.20
- USB Configurator 2.20 (new configuration)

Documentation

WICED Bluetooth Application



Quick Panel

Eclipse IDE for ModusToolbox®

Start
- New Application
- Search Online for Code Examples
- Search Online for Libraries and BSPs
- Refresh Quick Panel

Audio-20819EVB02 (CYW920819EVB-02)
- Build Audio-20819EVB02 Application
- Clean Audio-20819EVB02 Application

Launches
- Audio-20819EVB02 Attach_JLink
- Audio-20819EVB02 Attach_KitProg3_MiniProg4
- Audio-20819EVB02 Debug_JLink
- Audio-20819EVB02 Debug_KitProg3_MiniProg4
- Audio-20819EVB02 Program
- Generate Launches for Audio-20819EVB02

Tools
- Library Manager 1.2
- BTSpy
- Bluetooth Configurator 2.20 (new configuration)
- ClientControl
- Device Configurator 2.20 (new configuration)
- Power Estimator 1.2

Documentation
- WICED Bluetooth SDK Documentation

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 links to find Code Examples, Libraries, and BSPs.
- **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. For more information, see Use Configurators and Use Tools.
- **Documentation** – This may contain several documents in HTML format, which are included as part of the chosen BSP.

5. Configure Applications

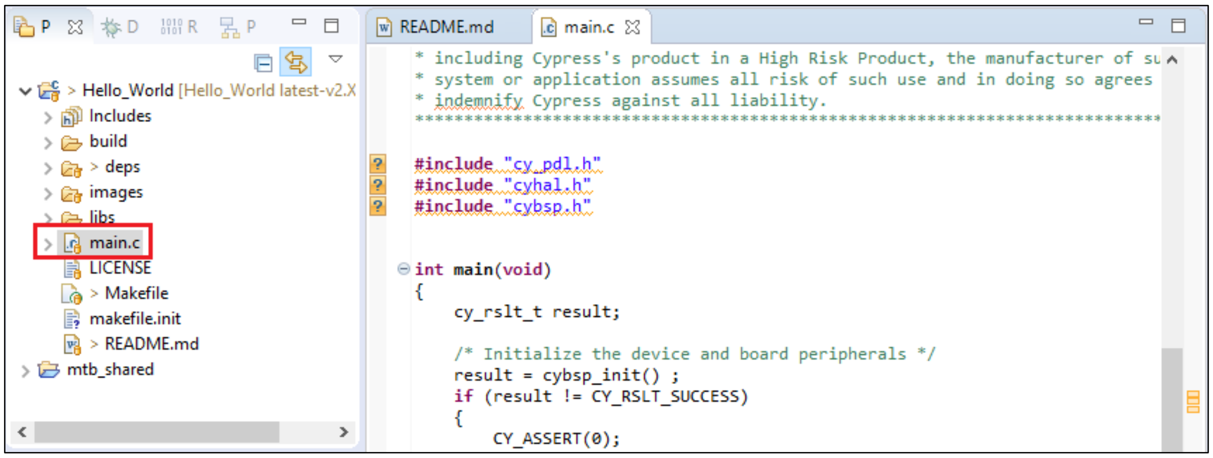
This chapter covers how to make various changes to your application. It includes:

- Modify Code
- Restore Shared Directory
- Use Configurators
- Use Tools
- Refresh Quick Panel
- Rename Application

Modify Code

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

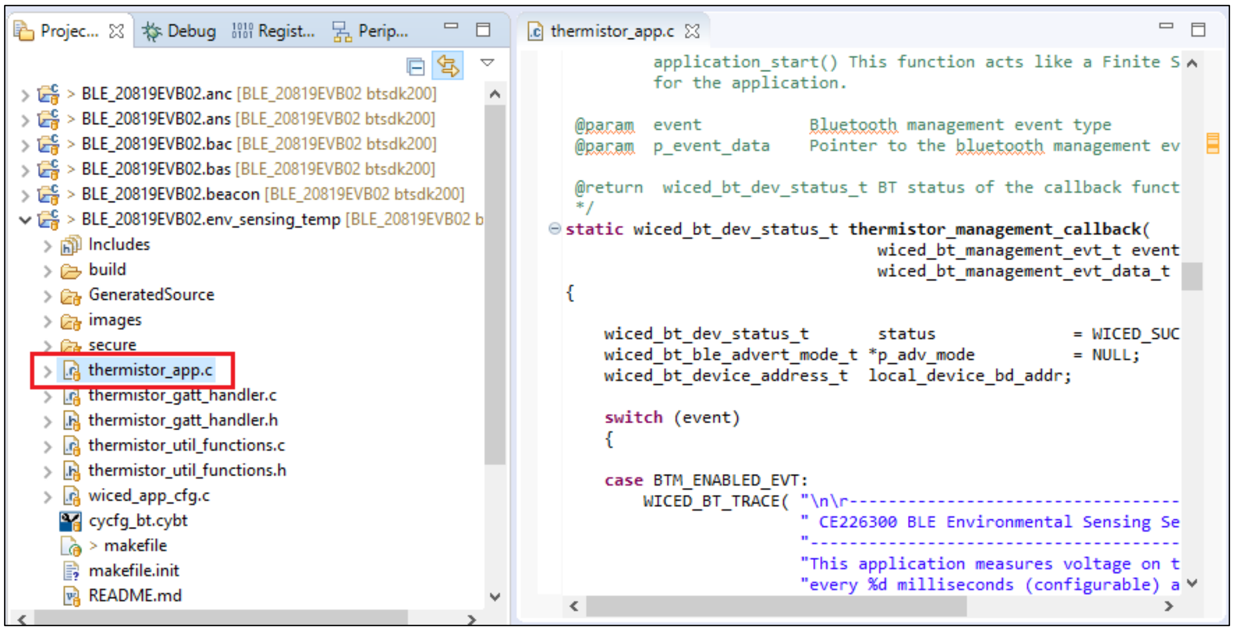
- **PSoC MCU:** In the Project Explorer, double-click the *main.c* file.



```
 * including Cypress's product in a High Risk Product, the manufacturer of su
 * system or application assumes all risk of such use and in doing so agrees
 * 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);
```

Note The includes in this example show “unresolved inclusions” because the files are located in the *mtb_shared* folder. They are resolved after a build.

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



```
> BLE_20819EVB02.anc [BLE_20819EVB02 btsdk200]
> BLE_20819EVB02.ans [BLE_20819EVB02 btsdk200]
> BLE_20819EVB02.bac [BLE_20819EVB02 btsdk200]
> BLE_20819EVB02.bas [BLE_20819EVB02 btsdk200]
> BLE_20819EVB02.beacon [BLE_20819EVB02 btsdk200]
> BLE_20819EVB02.env_sensing_temp [BLE_20819EVB02 b
    Includes
    build
    GeneratedSource
    images
    secure
    thermistor_app.c
    thermistor_gatt_handler.c
    thermistor_gatt_handler.h
    thermistor_util_functions.c
    thermistor_util_functions.h
    wiced_app_cfg.c
    cycfg_bt.cybt
    > makefile
    makefile.init
    README.md
```

```
        application_start() This function acts like a Finite S
        for the application.

 @param  event           Bluetooth management event type
 @param  p_event_data    Pointer to the bluetooth management ev

 @return  wiced_bt_dev_status_t BT status of the callback funct
 */
static wiced_bt_dev_status_t thermistor_management_callback(
                                wiced_bt_management_evt_t event
                                wiced_bt_management_evt_data_t

{

    wiced_bt_dev_status_t       status              = WICED_SUC
    wiced_bt_ble_advert_mode_t *p_adv_mode          = NULL;
    wiced_bt_device_address_t  local_device_bd_addr;

    switch (event)
    {

    case BTM_ENABLED_EVT:
        WICED_BT_TRACE( "\n\r-----------------------------------
                        " CE226300 BLE Environmental Sensing Se
                        "-----------------------------------
                        "This application measures voltage on t
                        "every %d milliseconds (configurable) a
```

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.

Restore Shared Directory

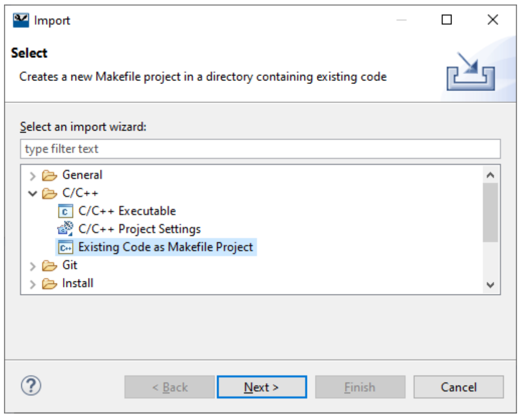
Beginning with the ModusToolbox 2.2 release, shared BSPs, libraries, and versions are located in a shared directory (named *mtb_shared*, by default) adjacent to your application directories. You can delete the *mtb_shared* directory at any time because it can be recreated. You might do this when sharing the application, for example. The shared directory only contains files that are already controlled and versioned, so you should NOT check it into a revision control system.

When using the Eclipse IDE, if you delete the shared library directory from disk and then regenerate it, the directory will not be restored properly. This is because several files required by the Eclipse IDE are not restored as they were when the application was created. To resolve this:

1. Regenerate the *mtb_shared* directory and assorted libraries on disk using `make getlibs` or the Library Manager.
2. In the Eclipse IDE, delete the "mtb_shared" folder shown in the Project Explorer.

   Note Do **NOT** select the check box "Delete project contents on disk" (if you do, you will have to regenerate it again).

3. Then, select **File > Import > C/C++ > Existing Code as Makefile Project** and click **Next >**



Import

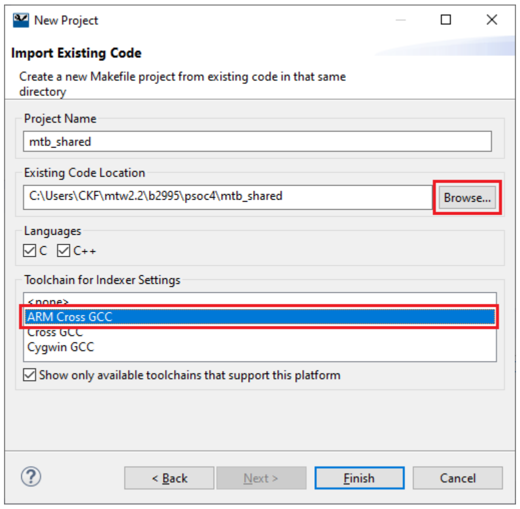
Select

Creates a new Makefile project in a directory containing existing code

Select an import wizard:

type filter text

4. On the Import Existing Code page.



New Project

Import Existing Code

Create a new Makefile project from existing code in that same directory

Project Name: mtb_shared

Existing Code Location: C:\Users\CKF\mtw2.2\b2995\psoc4\mtb_shared

Languages: ☑ C ☑ C++

Toolchain for Indexer Settings: <none>, ARM Cross GCC, Cross GCC, Cygwin GCC

☑ Show only available toolchains that support this platform

   a. Under **Existing Code Location**, click **Browse…**, navigate to the application's root directory, select the “mtb_shared” folder, and click **Select Folder**.

   b. Under **Toolchain for Indexer Settings**, select **ARM Cross GCC**.

   c. Click **Finish**.

5. After the import completes, build the application.

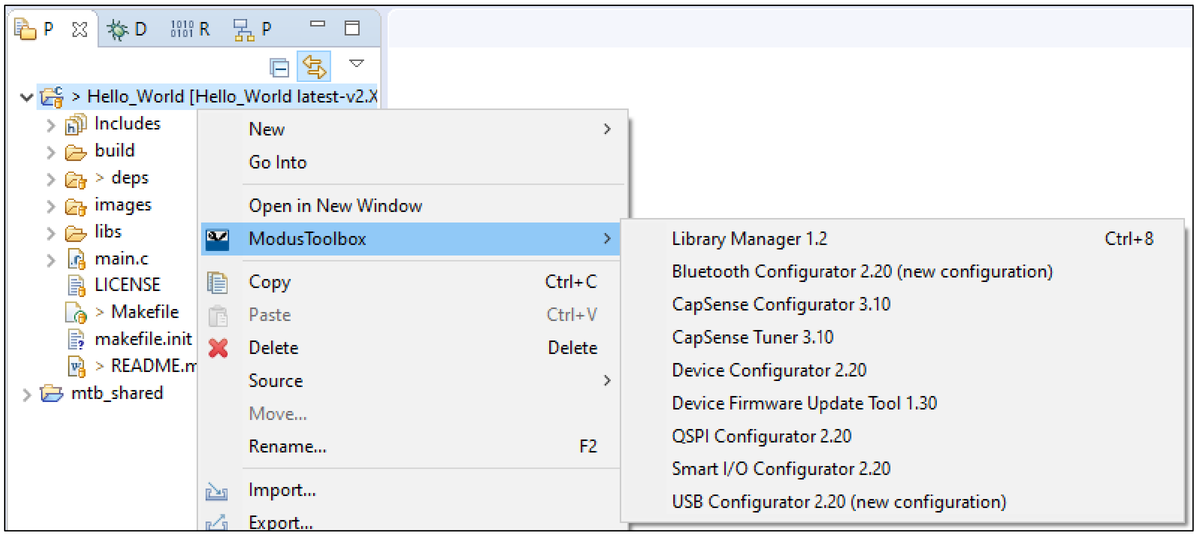
Use Configurators

ModusToolbox software provides graphical applications called configurators that make it easier to configure a hardware block. However, before you make changes to settings in configurators, you should first copy the configuration information to the application and override the BSP configuration or create a custom BSP. If you make changes to a standard BSP library, it will cause the repo to become dirty. Additionally, if the BSP is in the shared asset repository, changes will impact all applications that use the shared BSP. For more information, refer to the ModusToolbox User Guide.

Each configurator provides a separate guide, available from the configurator's **Help** menu.

Launching Configurators from the IDE

To launch a configurator from the Eclipse 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 Configurators without the IDE

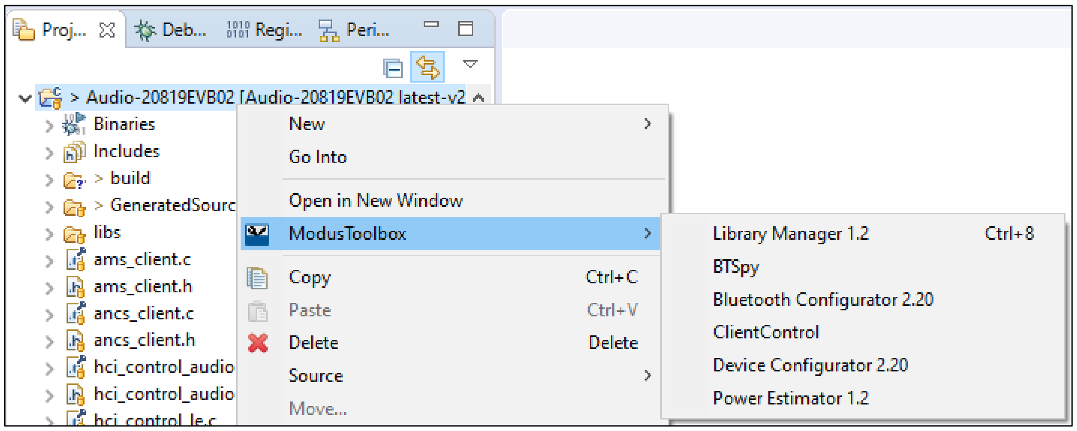
To launch any Configurator without using the IDE, navigate to the desired configurator's installation directory, and run the executable. Some configurators will open without any configuration information. You will need to open a configuration file or create a new one. Others will open with a default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

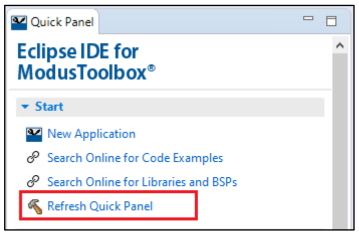
The Power Estimator tool provides an estimate of power consumed by a target device (also called platform). This is a stand-alone tool included with the ModusToolbox software. Currently, it can only be used with the following devices:

- 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.

Refresh Quick Panel

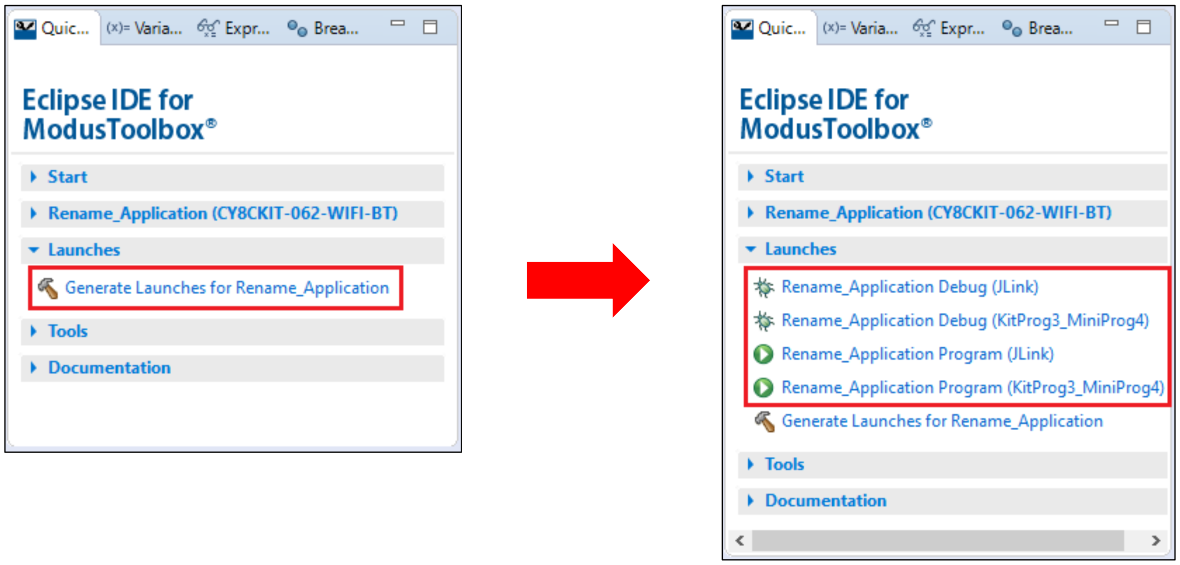
When you use tools external to the Eclipse IDE, such as the Library Manager or Device Configurator, it is likely that the Eclipse IDE will not refresh to detect changes made in those other tools. Use this link to refresh the Eclipse IDE. You may notice new documentation links or a new Active BSP in the Quick Panel as an indication that your application has new libraries and/or components.


Quick Panel
Eclipse IDE for ModusToolbox®
Start
New Application
Search Online for Code Examples
Search Online for Libraries and BSPs
Refresh Quick Panel

Rename Application

The Eclipse IDE for ModusToolbox uses the standard Eclipse rename functionality. That is, right-click on the application and select **Rename**. If you use the rename feature, you will need to update your application's launch configurations. The easiest way to do this is to use the "Generate Launches…" link in the Quick Panel.

After renaming the application, select it in the Project Explorer and notice that there is only one item under Launches. Click on the "Generate Launches…" link. After a few moments, the generate process completes. Click on the application again in the Project Explorer and notice that all the items are shown under Launches in the Quick Panel.


Quic... (x)= Varia... Expr... Brea...
Eclipse IDE for ModusToolbox®
Start
Rename_Application (CY8CKIT-062-WIFI-BT)
Launches
Generate Launches for Rename_Application
Tools
Documentation
Quic... (x)= Varia... Expr... Brea...
Eclipse IDE for ModusToolbox®
Start
Rename_Application (CY8CKIT-062-WIFI-BT)
Launches
Rename_Application Debug (JLink)
Rename_Application Debug (KitProg3_MiniProg4)
Rename_Application Program (JLink)
Rename_Application Program (KitProg3_MiniProg4)
Generate Launches for Rename_Application
Tools
Documentation

6. 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 Eclipse 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.

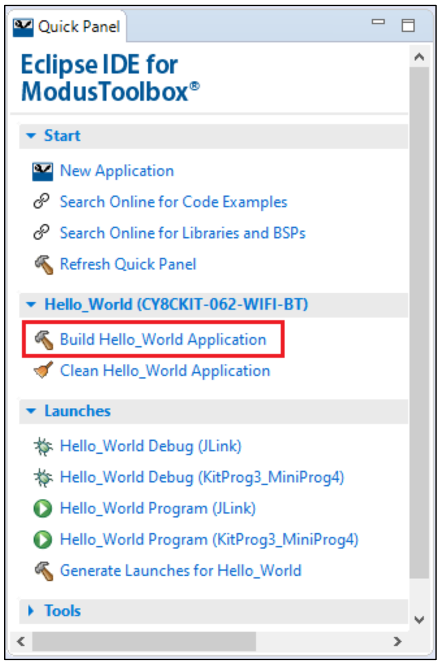
- Build with Make
- Build with IDE
- GCC Version

Build with Make

You can build applications from the command line using Make. Refer to the ModusToolbox User Guide (also located in the *<install>/ide_<version>/docs* directory) for more information. The document is also available from the IDE's **Help** menu.

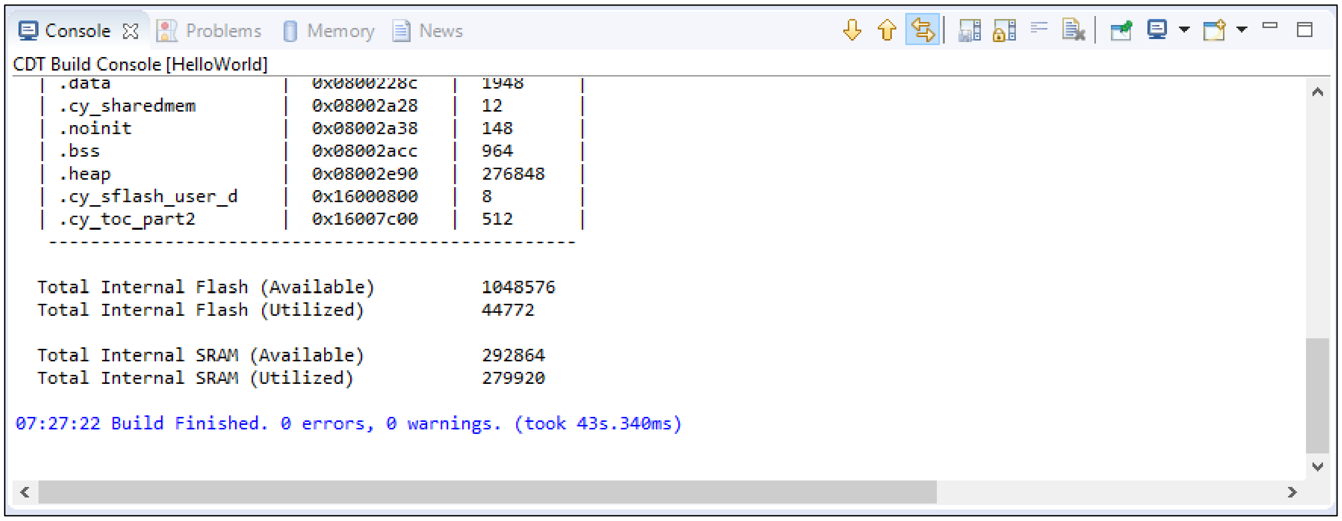
Build with IDE

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.


Quick Panel
Eclipse IDE for ModusToolbox®
Start
New Application
Search Online for Code Examples
Search Online for Libraries and BSPs
Refresh Quick Panel
Hello_World (CY8CKIT-062-WIFI-BT)
Build Hello_World Application
Clean Hello_World Application
Launches
Hello_World Debug (JLink)
Hello_World Debug (KitProg3_MiniProg4)
Hello_World Program (JLink)
Hello_World Program (KitProg3_MiniProg4)
Generate Launches for Hello_World
Tools

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.



```
  | .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.

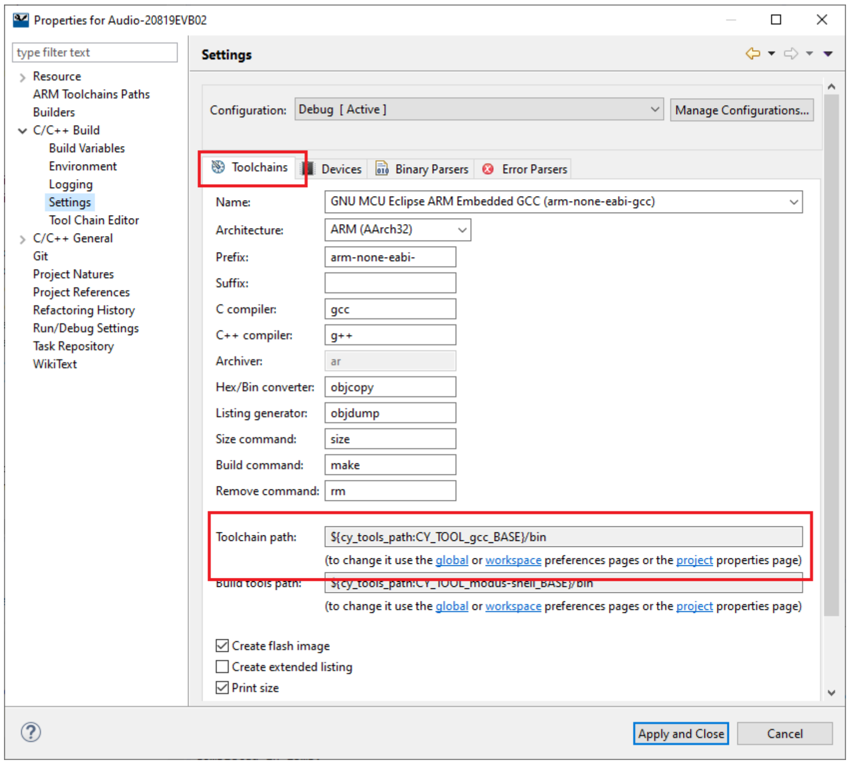
You can access the different views using the **Display Selected Console** button (or the pull-down arrow). The Global Build Console is the only way to see all of the results in one place. If you just have the standard Console open, it resets every time a new project in the application starts building. 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 Eclipse 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 9.3.1 as the preferred toolchain to use with the Eclipse 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.



Configuration: Debug [Active]

Name: GNU MCU Eclipse ARM Embedded GCC (arm-none-eabi-gcc)
Architecture: ARM (AArch32)
Prefix: arm-none-eabi-
Suffix:
C compiler: gcc
C++ compiler: g++
Archiver: ar
Hex/Bin converter: objcopy
Listing generator: objdump
Size command: size
Build command: make
Remove command: rm

Toolchain path: ${cy_tools_path:CY_TOOL_gcc_BASE}/bin
(to change it use the global or workspace preferences pages or the project properties page)
Build tools path: ${cy_tools_path:CY_TOOL_modus-shell_BASE}/bin
(to change it use the global or workspace preferences pages or the project properties page)

☑ Create flash image
☐ Create extended listing
☑ Print size

7. 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 Eclipse IDE for PSoC 6 devices and WICED Bluetooth devices.

PSoC MCU Programming/Debugging

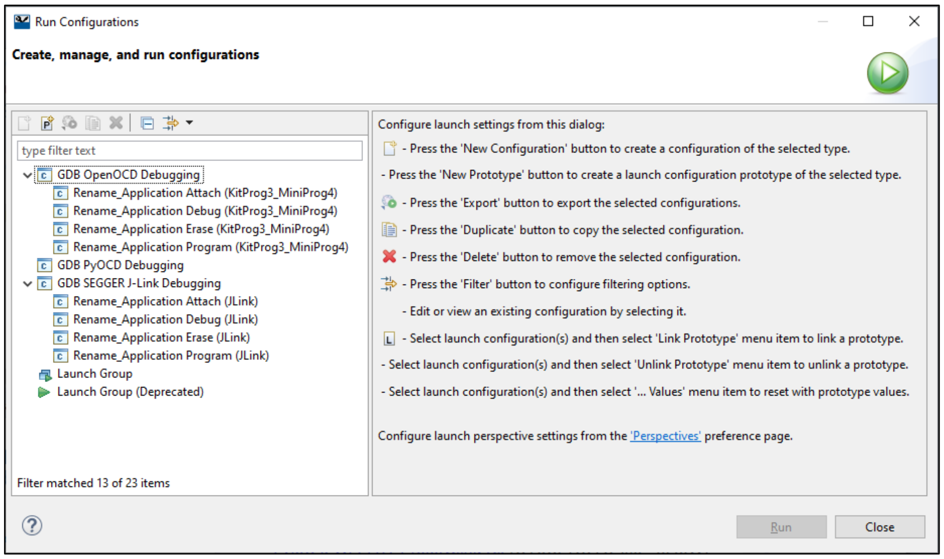
Launch Configurations

The flow for programming and debugging is similar for all devices. The Eclipse 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_MiniProg4 (included on-board on most Cypress PSoC 6 and PSoC 4 based kits) and another for SEGGER J-Link.

Note The KitProg3_MiniProg4 launch configuration also supports the MiniProg4, which you attach to the kit via a 10-pin debug connection. See the MiniProg4 product page for details. The MiniProg3 is not supported.

The Launch Configurations are shown in the Run/Debug Configurations dialog, similar to the following.



Run Configurations

Create, manage, and run configurations

type filter text

- GDB OpenOCD Debugging
  - Rename_Application Attach (KitProg3_MiniProg4)
  - Rename_Application Debug (KitProg3_MiniProg4)
  - Rename_Application Erase (KitProg3_MiniProg4)
  - Rename_Application Program (KitProg3_MiniProg4)
- GDB PyOCD Debugging
- GDB SEGGER J-Link Debugging
  - Rename_Application Attach (JLink)
  - Rename_Application Debug (JLink)
  - Rename_Application Erase (JLink)
  - Rename_Application Program (JLink)
- Launch Group
- Launch Group (Deprecated)

Filter matched 13 of 23 items

Configure launch settings from this dialog:

- Press the 'New Configuration' button to create a configuration of the selected type.
- Press the 'New Prototype' button to create a launch configuration prototype of the selected type.
- Press the 'Export' button to export the selected configurations.
- Press the 'Duplicate' button to copy the selected configuration.
- Press the 'Delete' button to remove the selected configuration.
- Press the 'Filter' button to configure filtering options.
- Edit or view an existing configuration by selecting it.
- Select launch configuration(s) and then select 'Link Prototype' menu item to link a prototype.
- Select launch configuration(s) and then select 'Unlink Prototype' menu item to unlink a prototype.
- Select launch configuration(s) and then select '... Values' menu item to reset with prototype values.

Configure launch perspective settings from the 'Perspectives' preference page.

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_MiniProg4)

CapSenseSlider Debug (JLink)

Note KitProg3_MiniProg4 configurations may not work if a J-Link probe is attached to the kit.

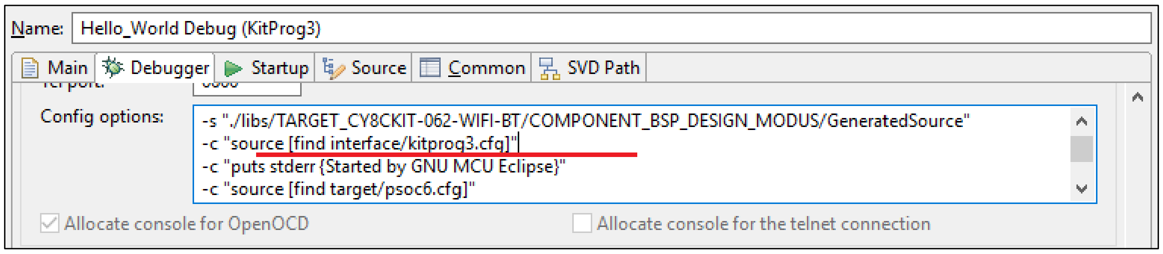
When an application is created, the tool generates the following launch configurations for both KitProg3_MiniProg4 and J-Link. Some items display in the **Quick Panel**, and some are in the Run/Debug Configurations dialog only.

- **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.

Using JTAG Interface in MiniProg4

The MiniProg4 can interact with the target via the SWD or JTAG protocol. By default, the SWD protocol will be used.

In order to use JTAG, you need to add specific commands for the appropriate Debug Configuration. Under the **Debugger** tab in the **Config options** field (after -c “source [find interface/kitprog3.cfg]”), insert the following lines:


Name: Hello_World Debug (KitProg3)
Main
Debugger
Startup
Source
Common
SVD Path
Config options:
Allocate console for OpenOCD
Allocate console for the telnet connection

```
-s "./libs/TARGET_CY8CKIT-062-WIFI-BT/COMPONENT_BSP_DESIGN_MODUS/GeneratedSource"
-c "source [find interface/kitprog3.cfg]"
-c "puts stderr {Started by GNU MCU Eclipse}"
-c "source [find target/psoc6.cfg]"
```

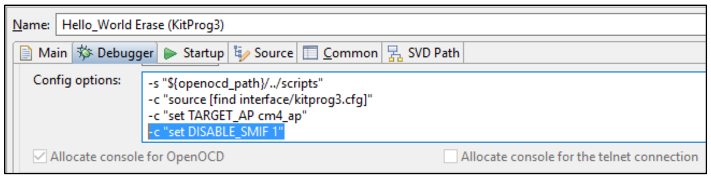
- -c “cmsis_dap_vid_pid 0x04B4 0xF151 0x04B4 0xF152”
- -c “transport select jtag”

Erasing external memory

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

For **PSoC 64 Secure Boot kits**, such as CY8CKIT-064S0S2-4343W and CY8CPROTO-064S1-SB, enter the following to disable external memory programming via SMIF:

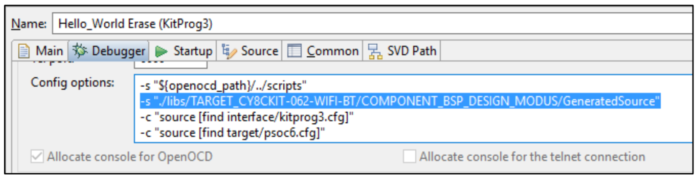
```
-c “set DISABLE_SMIF 1”
```


Name: Hello_World Erase (KitProg3)
Main
Debugger
Startup
Source
Common
SVD Path
Config options:
Allocate console for OpenOCD
Allocate console for the telnet connection

```
-s "${openocd_path}/../scripts"
-c "source [find interface/kitprog3.cfg]"
-c "set TARGET_AP cm4_ap"
-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”
```


Name: Hello_World Erase (KitProg3)
Main
Debugger
Startup
Source
Common
SVD Path
Config options:
Allocate console for OpenOCD
Allocate console for the telnet connection

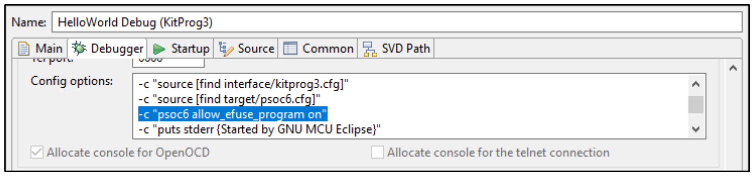
```
-s "${openocd_path}/../scripts"
-s "./libs/TARGET_CY8CKIT-062-WIFI-BT/COMPONENT_BSP_DESIGN_MODUS/GeneratedSource"
-c "source [find interface/kitprog3.cfg]"
-c "source [find target/psoc6.cfg]"
```

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"
```


Name: HelloWorld Debug (KitProg3)

Config options:

```
-c "source [find interface/kitprog3.cfg]"
-c "source [find target/psoc6.cfg]"
-c "psoc6 allow_efuse_program on"
-c "puts stderr {Started by GNU MCU Eclipse}"
```

☑ Allocate console for OpenOCD ☐ Allocate console for the telnet connection

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 Eclipse IDE includes launch configurations set up for two probes: Cypress KitProg3 (built into Cypress kits) and Segger J-Link.

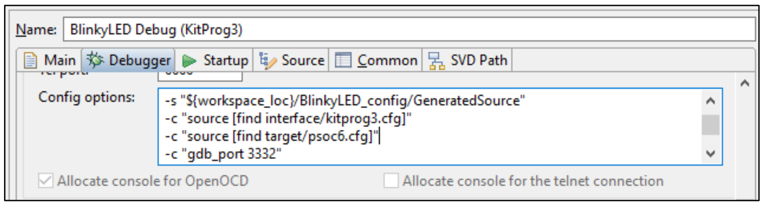
Communication Firmware	Debug Connection	More Information
Cypress-provided KitProg3	OpenOCD via CMSIS-DAP	https://www.cypress.com/products/psoc-programming-solutions
SEGGER-provided J-Link DLL	SEGGER J-Link	SEGGER J-Link

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]")


Name: BlinkyLED Debug (KitProg3)

Config options:

```
-s "${workspace_loc}/BlinkyLED_config/GeneratedSource"
-c "source [find interface/kitprog3.cfg]"
-c "source [find target/psoc6.cfg]"
-c "gdb_port 3332"
```

☑ Allocate console for OpenOCD ☐ Allocate console for the telnet connection

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
  ```

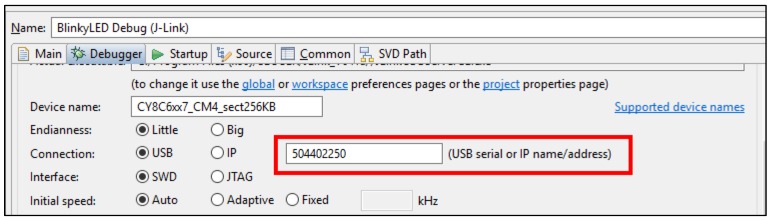
Selecting by both VID/PID and Serial Number

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:



Name: BlinkyLED Debug (J-Link)

(to change it use the global or workspace preferences pages or the project properties page)

Device name: CY8C6xx7_CM4_sect256KB

Endianness: [✓ Little] Big

Connection: [✓ USB] IP 504402250 (USB serial or IP name/address)

Interface: [✓ SWD] JTAG

Initial speed: [✓ Auto] Adaptive Fixed kHz

KitProg Firmware Loader

The PSoC 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 (Bulk) protocol by default, which is up to ~2.5 times faster than the CMSIS-DAP (HID) protocol. 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:

<install_path>\ModusToolbox\tools_<version>\fw-loader\bin\

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. The fw-loader tool also provides a readme text file in the fw-loader installation directory.

PSoC 4 Flash Security Programming

PSoC 4 devices include a flexible flash-protection system that controls access to flash memory. This feature secures proprietary code, but it can also be used to protect against inadvertent writes to the bootloader portion of flash. Refer to the device's Architecture Technical Reference Manual for details.

Flash consists of rows (64/128/256 bytes, depending on the PSoC 4 device). Each row of flash can have its protection level independently set. You can assign one of two protection levels to each row, as shown in the following table:

Protection Setting	Allowed	Not Allowed
Unprotected	External read and write, Internal read and write	–
Full Protection	External read, Internal read	External write, Internal write

To apply flash security, include the appropriate data in the application by creating an array of *num_rows* / 8 size, where *num_rows* is the actual number of flash rows on the device. Each byte of this array is responsible for protection of 8 flash rows, and it sets one of two protection levels:

- 0 – unprotected
- 1 – full protection

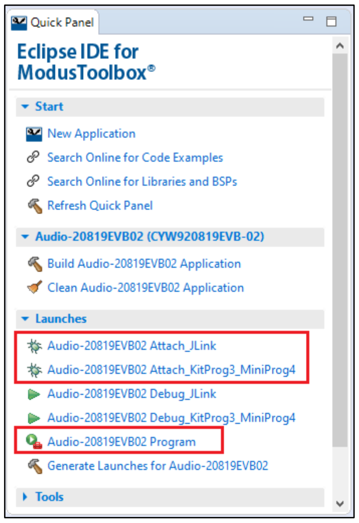
For example, to protect the first 25 rows and the last 5 out of 256 rows, include the following array in your application:

```
CY_SECTION(".cyflashprotect") __USED
static const unsigned char flash_protect[0x20] =
{
  0xFF, 0xFF, 0xFF, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00
};
```

To enable flash security, program the device with the updated code. If you wish to reprogram a protected device, you must first erase it.

Bluetooth Programming/Debugging

WICED Bluetooth applications have different launch configurations than PSoC 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.


Quick Panel
Eclipse IDE for ModusToolbox®
Start
New Application
Search Online for Code Examples
Search Online for Libraries and BSPs
Refresh Quick Panel
Audio-20819EVB02 (CYW920819EVB-02)
Build Audio-20819EVB02 Application
Clean Audio-20819EVB02 Application
Launches
Audio-20819EVB02 Attach_JLink
Audio-20819EVB02 Attach_KitProg3_MiniProg4
Audio-20819EVB02 Debug_JLink
Audio-20819EVB02 Debug_KitProg3_MiniProg4
Audio-20819EVB02 Program
Generate Launches for Audio-20819EVB02
Tools

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_MiniProg4**

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 Eclipse 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 for more details.

Mbed OS Programming/Debugging

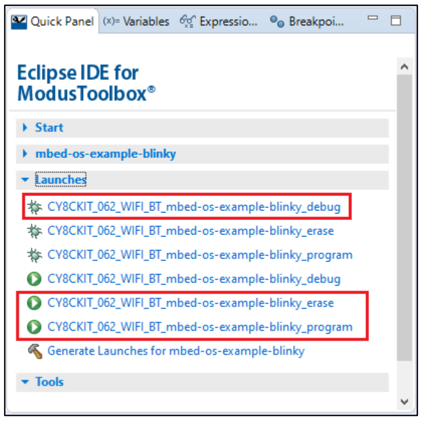
Launch Configurations

When an application is imported from Mbed OS into the Eclipse 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.


Quick Panel
(x)= Variables
Expressio...
Breakpoi...
Eclipse IDE for ModusToolbox®
Start
mbed-os-example-blinky
Launches
CY8CKIT_062_WIFI_BT_mbed-os-example-blinky_debug
CY8CKIT_062_WIFI_BT_mbed-os-example-blinky_erase
CY8CKIT_062_WIFI_BT_mbed-os-example-blinky_program
CY8CKIT_062_WIFI_BT_mbed-os-example-blinky_debug
CY8CKIT_062_WIFI_BT_mbed-os-example-blinky_erase
CY8CKIT_062_WIFI_BT_mbed-os-example-blinky_program
Generate Launches for mbed-os-example-blinky
Tools

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:

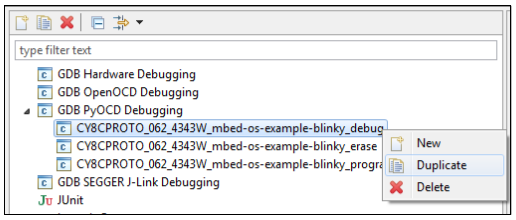
1. Open the Preferences dialog and select **Run/Debug > Console**.
2. Click on the option for **Standard Error text color**, select the desired color, and click **OK**.
3. Then, click **Apply and Close** to close the Preferences dialog.

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.

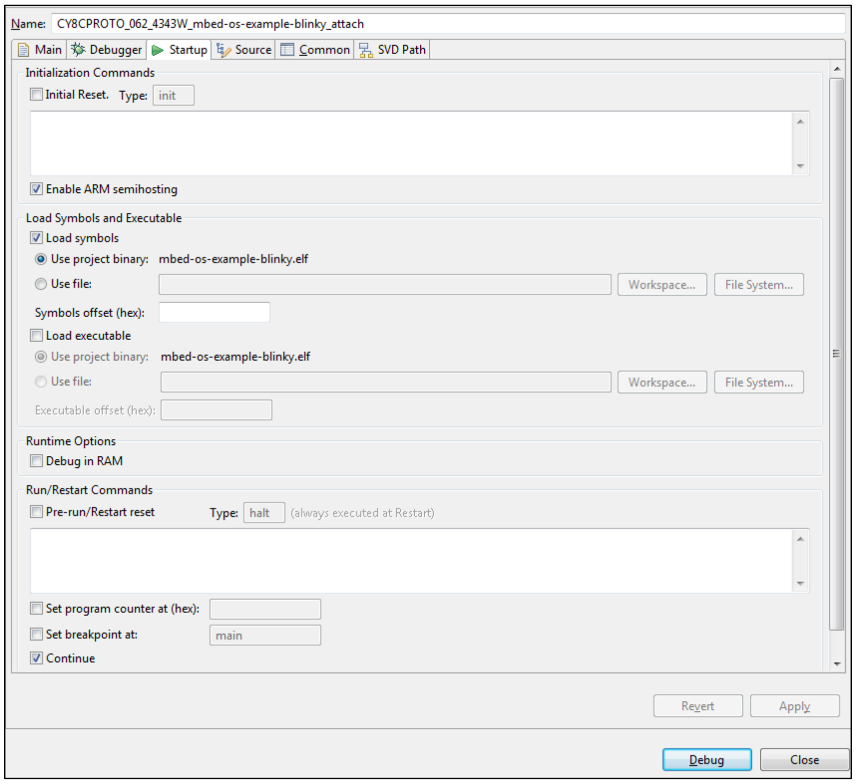
1. See Mbed OS to ModusToolbox Flow for instructions to import an example Mbed project into the Eclipse IDE workspace.

2. Open the Debug Configurations dialog, and duplicate the Debug configuration; for example, “CY8CPROTO_062_4343W_mbed-os-example-blinky_debug.”


type filter text
GDB Hardware Debugging
GDB OpenOCD Debugging
GDB PyOCD Debugging
CY8CPROTO_062_4343W_mbed-os-example-blinky_debug
CY8CPROTO_062_4343W_mbed-os-example-blinky_erase
CY8CPROTO_062_4343W_mbed-os-example-blinky_progra
GDB SEGGER J-Link Debugging
JUnit
New
Duplicate
Delete

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.



Name: CY8CPROTO_062_4343W_mbed-os-example-blinky_attach

Initialization Commands

- [ ] Initial Reset. Type: init
- [✓] Enable ARM semihosting

Load Symbols and Executable

- [✓] Load symbols
- (●) Use project binary: mbed-os-example-blinky.elf
- ( ) Use file:
- Symbols offset (hex):
- [ ] Load executable
- (●) Use project binary: mbed-os-example-blinky.elf
- ( ) Use file:
- Executable offset (hex):

Runtime Options

- [ ] Debug in RAM

Run/Restart Commands

- [ ] Pre-run/Restart reset Type: halt (always executed at Restart)
- [ ] Set program counter at (hex):
- [ ] Set breakpoint at: main
- [✓] Continue

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.