Eclipse IDE for ModusToolbox™ user guide

ModusToolbox™ tools package version 3.1.0

About this document

Scope and purpose

ModusToolbox™ software is a set of tools that enable you to integrate our 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.

This document provides information about creating applications as well as building, programming, and debugging them using the Eclipse IDE.

Document conventions

Convention	Explanation
Bold	Emphasizes heading levels, column headings, menus and sub-menus
Italics	Denotes file names and paths
`Courier New`	Denotes APIs, functions, interrupt handlers, events, data types, error handlers, file/folder names, directories, command line inputs, code snippets
File > New	Indicates that a cascading sub-menu opens when you select a menu item

Reference documents

Refer to the following documents for more information as needed:

- [ModusToolbox™ tools package installation guide](#)
- [ModusToolbox™ tools package user guide](#)
- [Eclipse survival guide](#)

Table of contents

1 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](#)
- [Open Project Creator tool](#)
- [Create an application](#)
- [Build application](#)
- [Program application](#)
- [Export/share application](#)
- [Import application](#)
- [Search online for code examples](#)
- [Search online for libraries/BSPs](#)
- [Access training material](#)

1.1 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™ tools package 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.
- You can also launch Eclipse IDE from the dashboard for all operating systems. For more information refer to the [dashboard user guide](#).

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 [Eclipse survival guide](#).
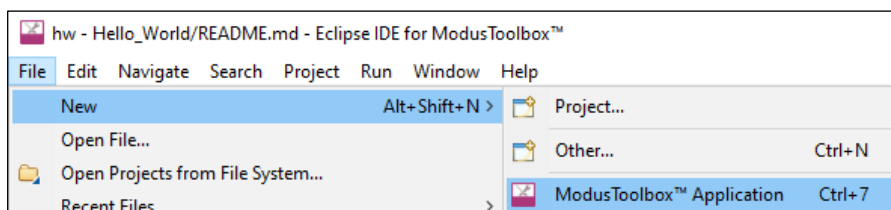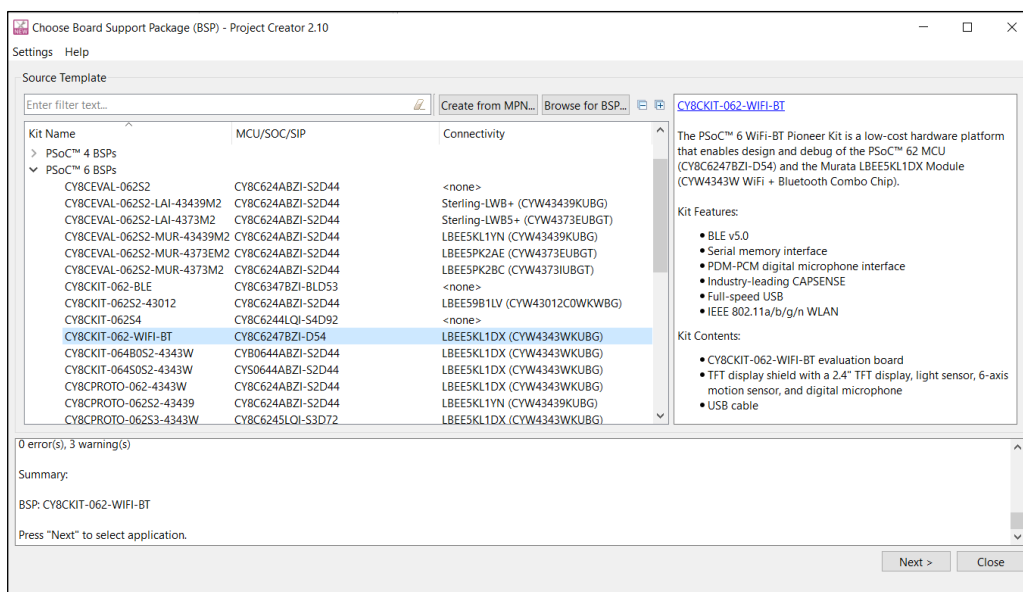
1.2   Open Project Creator tool

Click the **New Application** link in the Eclipse IDE Quick Panel.
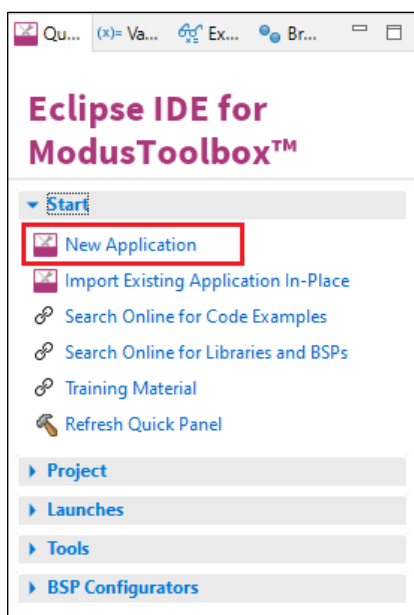
You can also select **File > New > ModusToolbox™ Application**.

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

For more details about using this tool, refer to the Project Creator user guide.

1.3 Create an application

This section provides a walkthrough for creating a ModusToolbox™ application.

1.3.1 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, including PSoC™ 4, PSoC™ 6, and AIROC™ Bluetooth® BSPs. For this example, select the CY8CKIT-062-WIFI-BT kit.
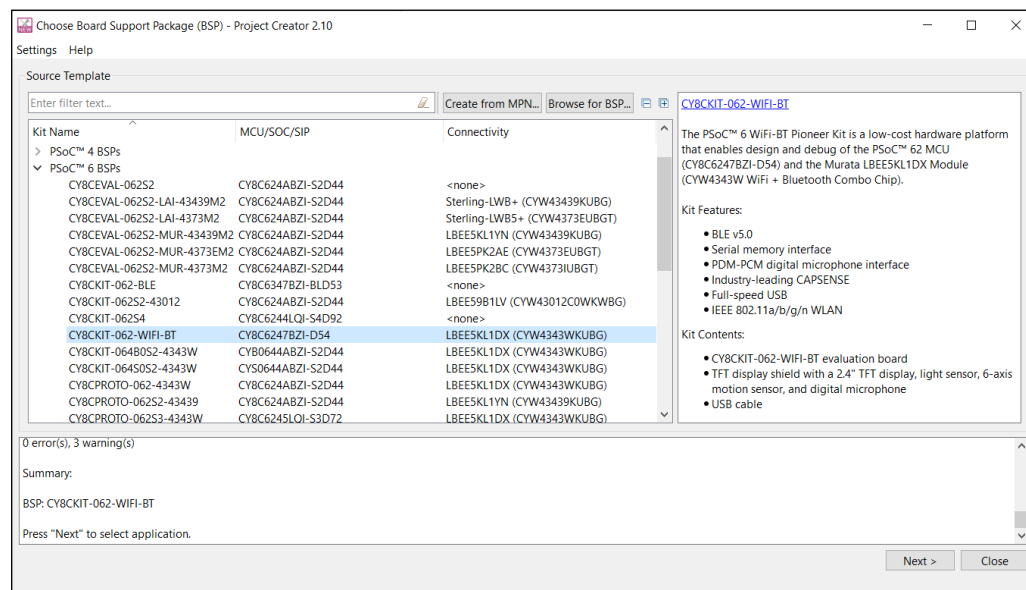
1.3.2 Select application

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.
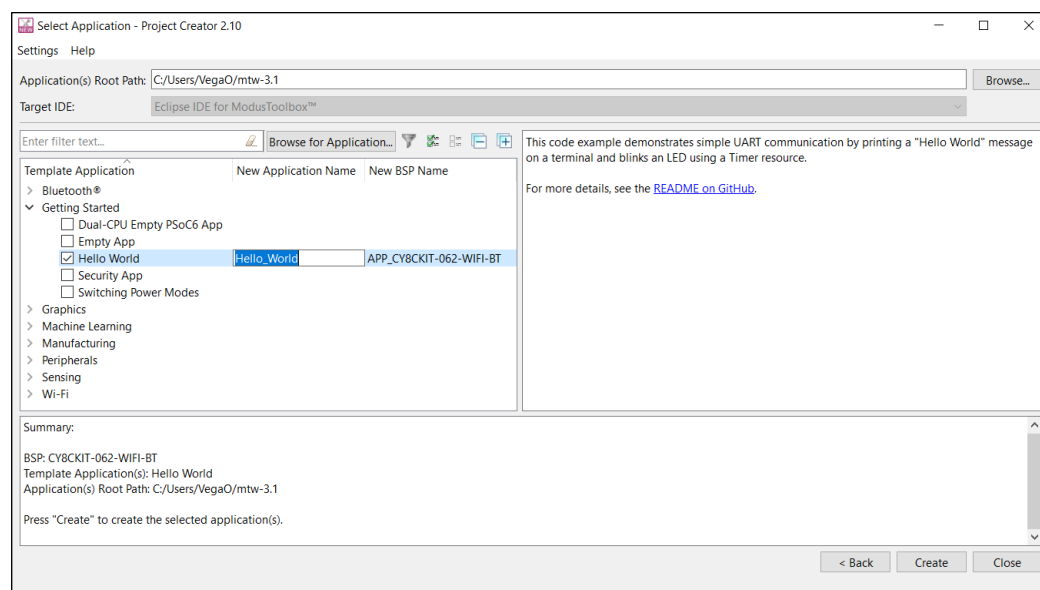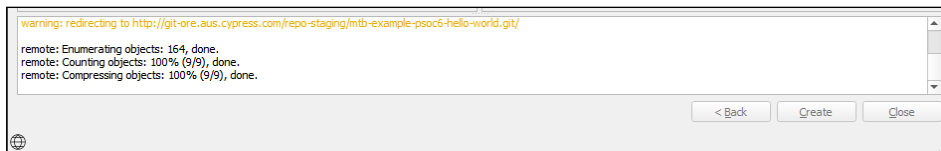
Getting started

For this example:

- Select the check box next to the "Hello World" application.
- If desired, type a name for the application and/or BSP. Do not use spaces.
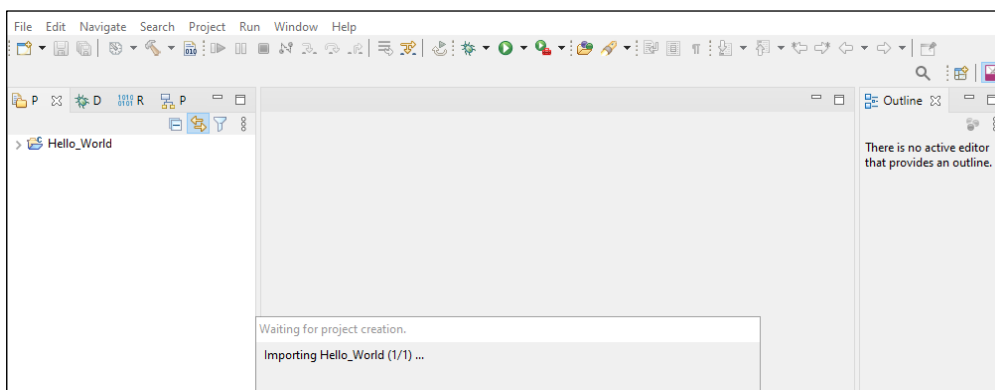
1.3.3   Create application

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

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

When complete, the Project Creator tool closes automatically if there are no errors or warnings. If there are warnings only, click close and the application will open in the IDE. In the IDE, a message will display about importing the project:

If there are errors, project creation will fail.

Getting started

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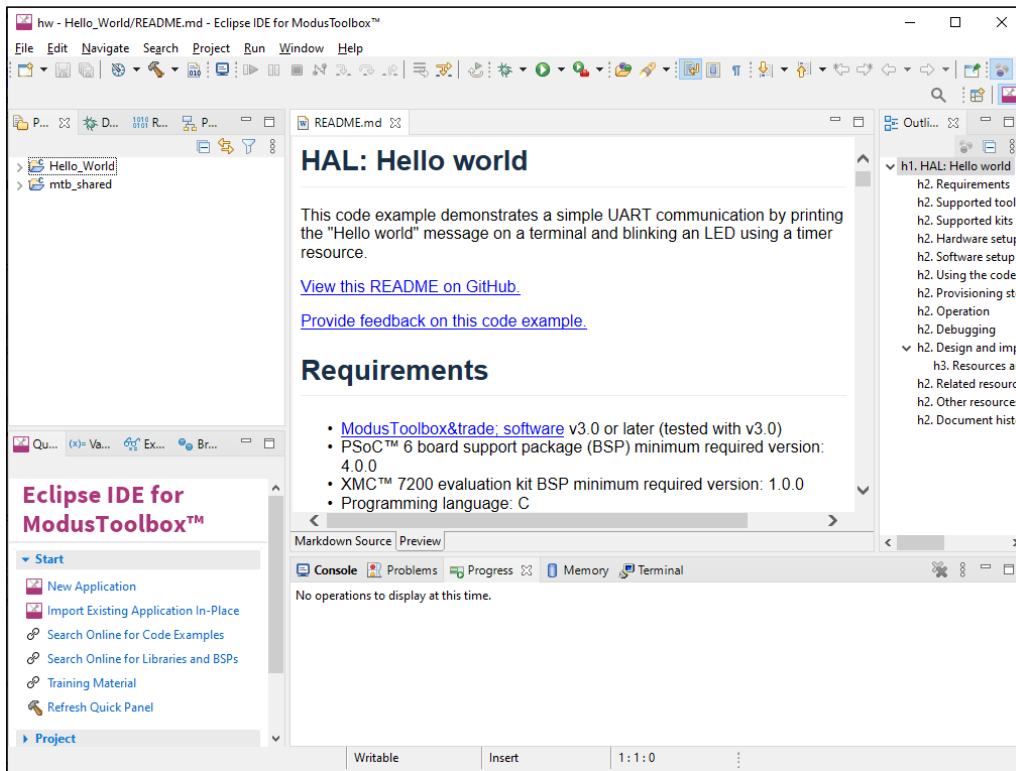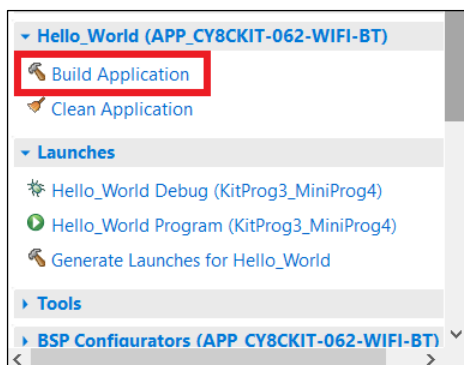

Note: *Many AIROC™ Bluetooth® applications contain multiple projects. For example, the BLE-20819EVB02 application contains projects for Bluetooth® Low Energy services such as anc, ans, bac, bas, beacon, etc.*
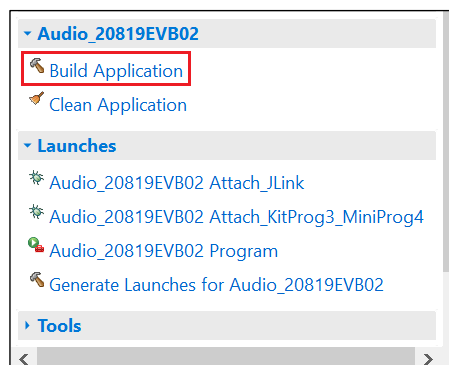
1.4   Build application

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 an AIROC™ Bluetooth® application.

PSoC™ MCU Application

AIROC™ Bluetooth® Application

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.

1.5 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 an AIROC™ Bluetooth® application.

PSoC™ MCU Application
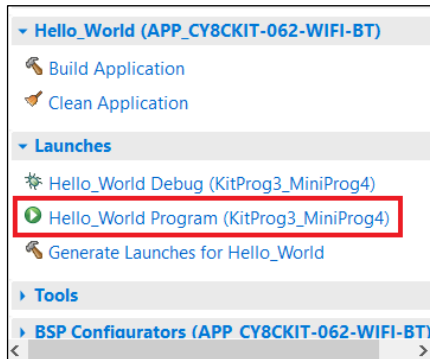
AIROC™ Bluetooth® Application
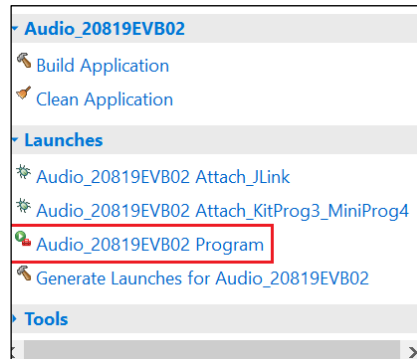
1.6 Export/share application

There are several ways to share a ModusToolbox™ application. Refer to the [ModusToolbox™ tools package user guide](#) in the "Using applications with third-party tools" chapter for more general details.

Using the Eclipse IDE, you can use the export function to essentially make a copy of your application that you can share with a colleague. There are several ways to export using Eclipse. The option we recommend is to use **Export > General > Archive File**, which creates a ZIP file.
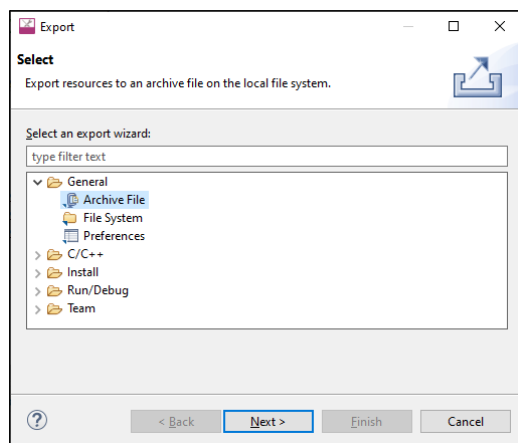

Keep in mind that a typical ModusToolbox™ application includes many libraries that are available on GitHub. These get regenerated during the `make getlibs` command. Therefore, you can substantially reduce the size of your exported application by excluding the libraries. Plus, the exported *mtb_shared* folder might retain hard-coded paths that would cause issues after importing the application.

For multi-core applications, the Eclipse IDE export function shows the core project folders along side the main application folder. These project folders are included in that main application folder, so you should exclude them from the export process as well.

On the Export dialog, select only the main application folder(s) to export. You may need to review your application in Eclipse to determine which folders you should exclude. By default, subproject folders include an extension after the application name.
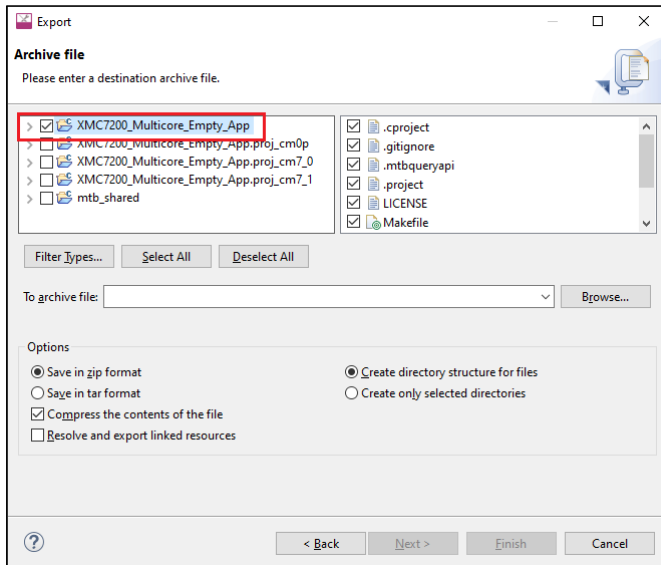
Click **Browse…** to navigate to the location to save the archive and provide a file name.

Click **Finish** to generate the archive file.

1.7 Import application

If you have a ModusToolbox™ application that you want to import into the Eclipse IDE, run the `make getlibs` command or use the Library Manager **Update** button on the application before importing it into Eclipse.

Note: *If the application is included in a ZIP file, you will need to extract it before running various commands.*
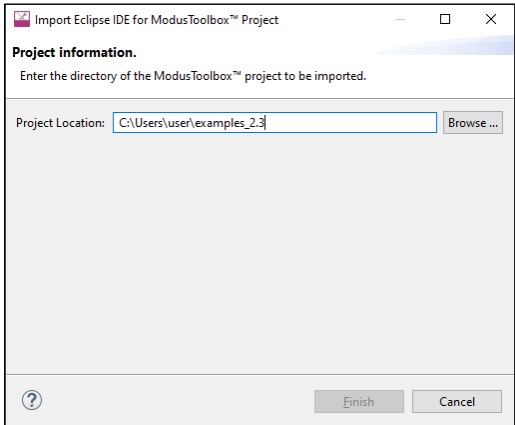
Then, use the **Import Existing Application In-Place** link on the Quick panel.

Note: *This option does not copy or move the application location on disk; it only makes a reference for Eclipse to the current application location.*
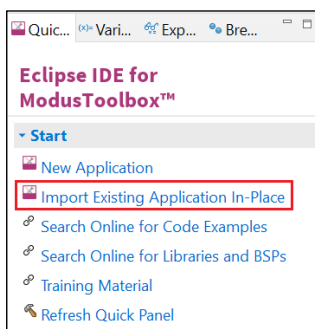
On the next page, click the **Browse…** button, navigate to the application directory, and click **Select Folder**.


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.*

1.8   Search online for code examples

Infineon 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. This opens a web browser to the GitHub repository to select and download the appropriate examples.
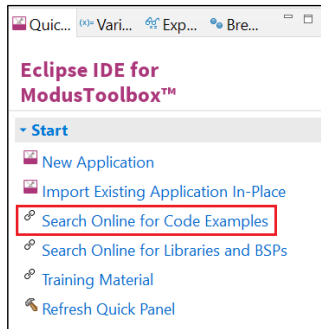
Code examples for ModusToolbox™ software

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™ tools package user guide.
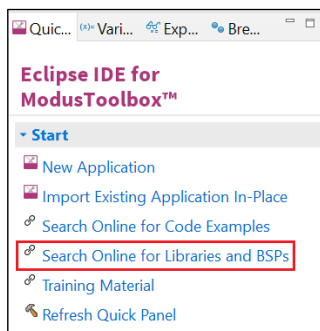
See the individual Amazon FreeRTOS example repository for details on how to import each code example into Amazon FreeRTOS build system respectively. Typically, you'll find the precise command line in the readme file.

Repo	Description
PSoC™ 6 MCU examples	This search link displays all PSoC™ 6 MCU examples for ModusToolbox™ software. These examples demonstrate the MCU and its features and functionality.
PSoC™ 4 MCU examples	This search link displays all PSoC™ 4 MCU examples for ModusToolbox™ software. These examples demonstrate the MCU and its features and functionality.
XMC MCU examples	This search link displays all XMC MCU examples for ModusToolbox™ software. These examples demonstrate the MCU and its features and functionality.
PMG1 MCU examples	This search link displays all PMG1 MCU examples for ModusToolbox™ software. These examples demonstrate the MCU and its features and functionality.

1.9 Search online for libraries/BSPs

Infineon 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. This opens a web browser to the GitHub repository that shows the ModusToolbox™ software page.

1.10 Access training material

Infineon also provides training material at GitHub. The **Quick Panel** provides a link to access these. Click the **Training Material** link. This opens a web browser to the GitHub repository that shows the ModusToolbox™ training page.

2   IDE description

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. We also provide 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," and adds tabs to the Project Explorer. "Perspective" is an Eclipse term for the initial set and layout of views in the IDE.
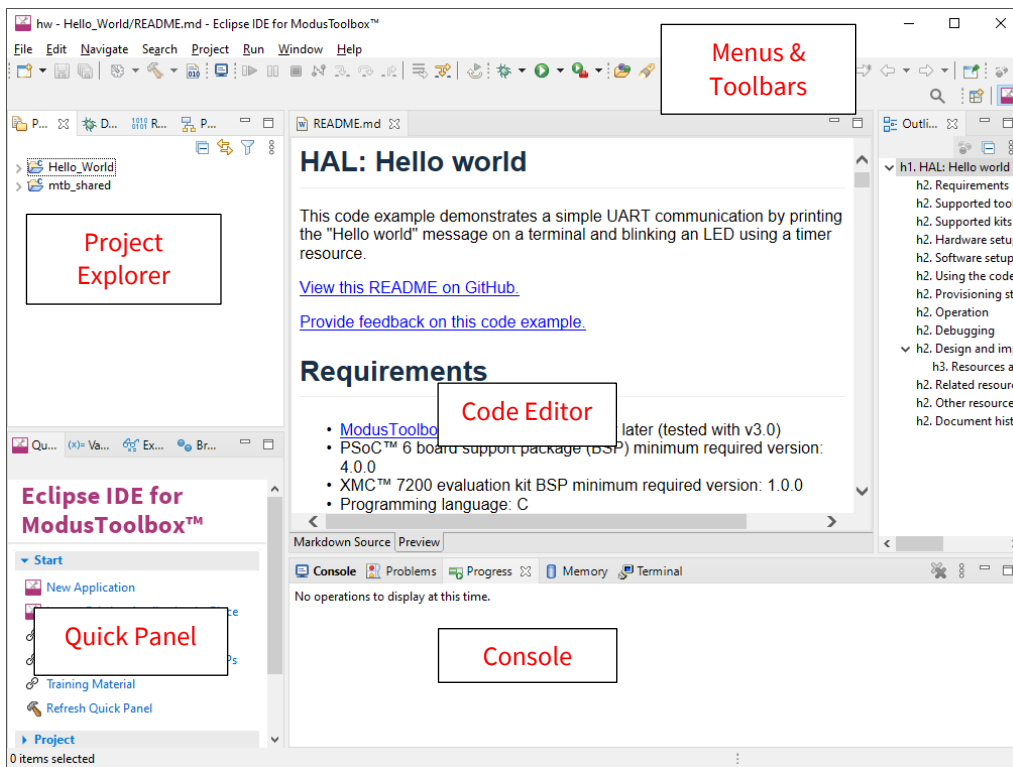
Note: *If you switch to a different perspective, you can restore the ModusToolbox™ Perspective by clicking the ModusToolbox™ icon button in the upper-right corner. You can also select **Perspective > Open Perspective** from the **Window** menu. To restore the ModusToolbox™ Perspective to the original 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.

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

- Console – Use these tools to review messages and access the integrated terminal.

IDE description

2.1 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 an AIROC™ Bluetooth® application.

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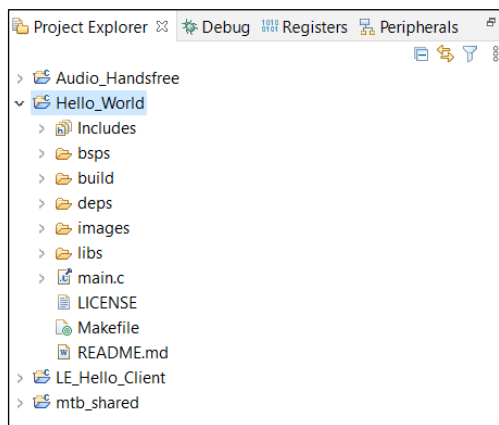
AIROC™ Bluetooth® Application
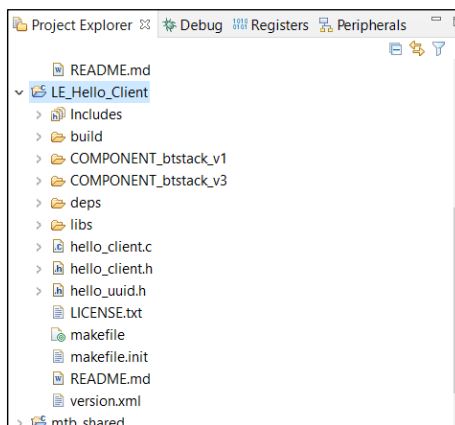
Both types of applications contain a similar project structure. Each contains the main application source code, and a Makefile. Note that PSoC™ MCU applications contain a libs directory, while AIROC™ Bluetooth® applications have a wiced_btsdk project with shared SDK, BSPs, and libraries for all applications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

2.2 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
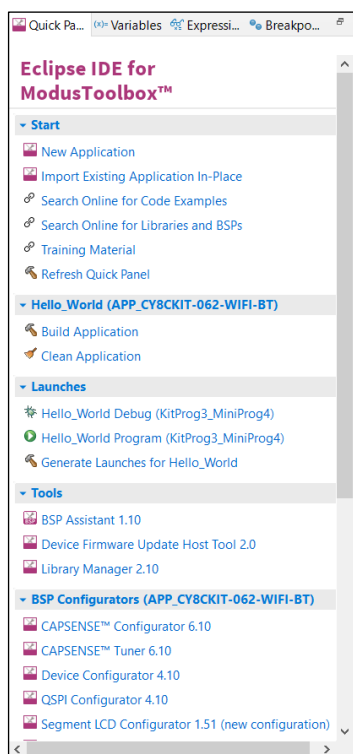
AIROC™ Bluetooth® Application
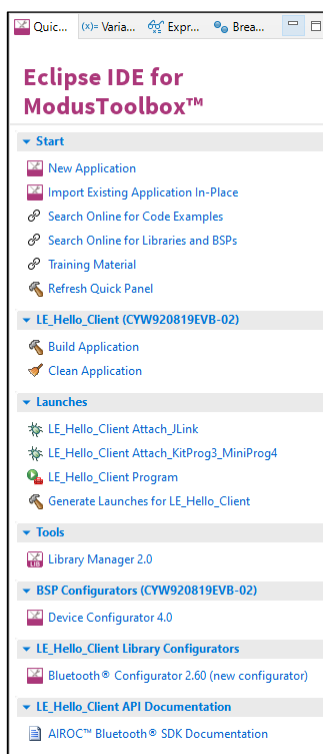
IDE description

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, BSPs, and training material.
- **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.

3   Configure applications

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

- [Modify code](#)
- [Use configurators](#)
- [Use tools](#)
- [Use integrated terminal](#)
- [Refresh Quick Panel](#)
- [Rename single-core application](#)
- [Restore shared directory](#)

3.1   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 them. However, if you want to update and change an 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.
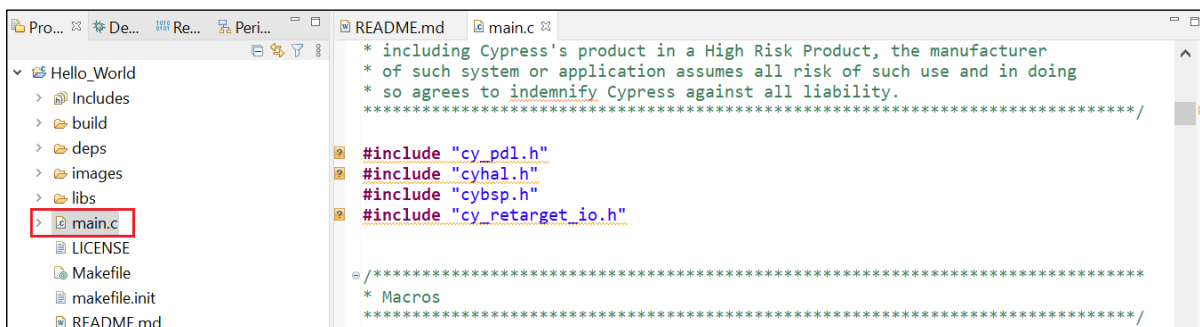
Note: *The "includes" in this example show "unresolved inclusions" because the files are located in the mtb_shared folder. This and other issues with IntelliSense are resolved after a build.*

- **AIROC™ Bluetooth®:** In the Project Explorer, expand the *<app-name>* project folder and double-click the application *<app-name>.c* file.
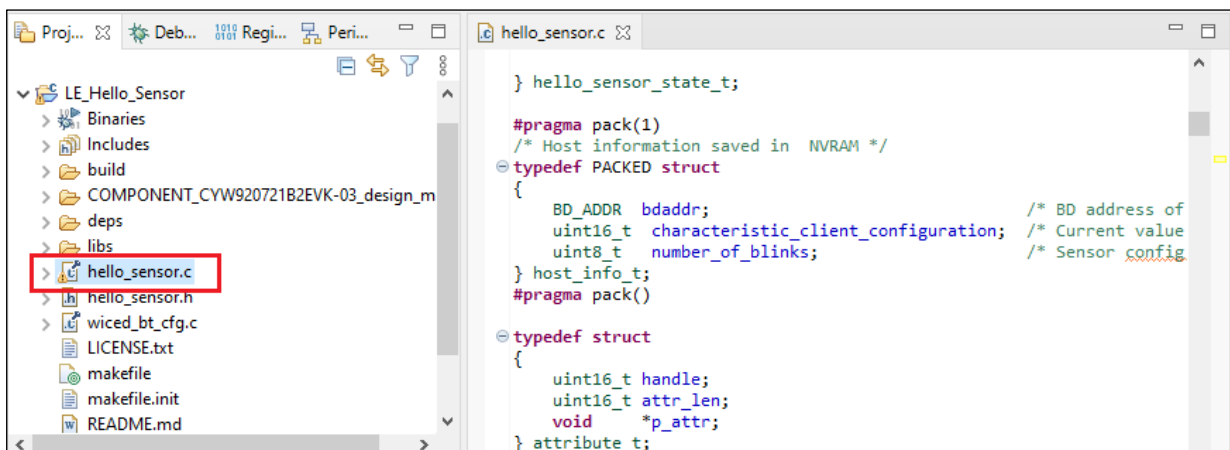
As you type into the file, an asterisk (*) will appear in the file's tab to indicate changes were made. The **Save/Save As** commands will also become available to select.

3.2 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™ tools package user guide.

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

3.2.1 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.
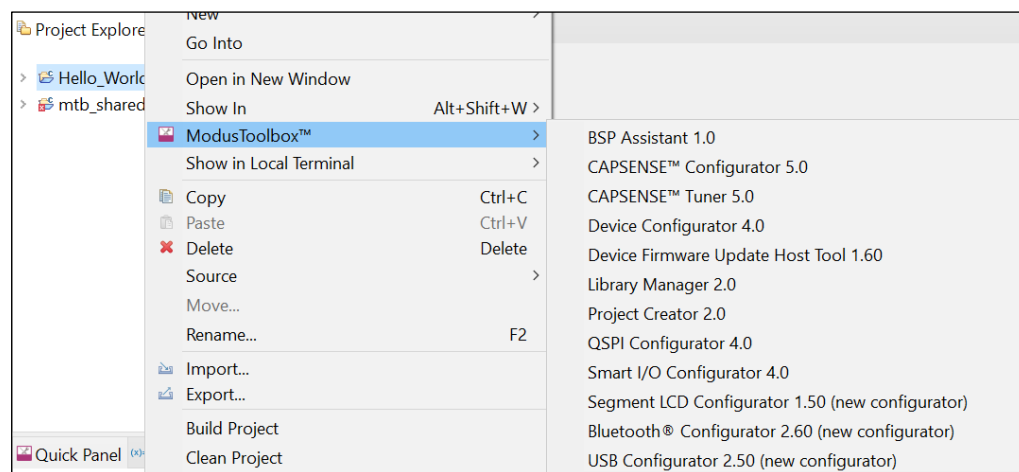

Note: *You can also launch available configurators from links in the Quick Panel.*

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.

3.2.2 Launching configurators without the IDE

To launch any Configurator without using the IDE, refer to the applicable configurator guide for details about configuration information. You may need to open a configuration file or create a new one.

3.3 Use tools

In addition to the configurators, there are several tools, including Library Manager, BTSpy, ClientControl, 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.

3.3.1 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.
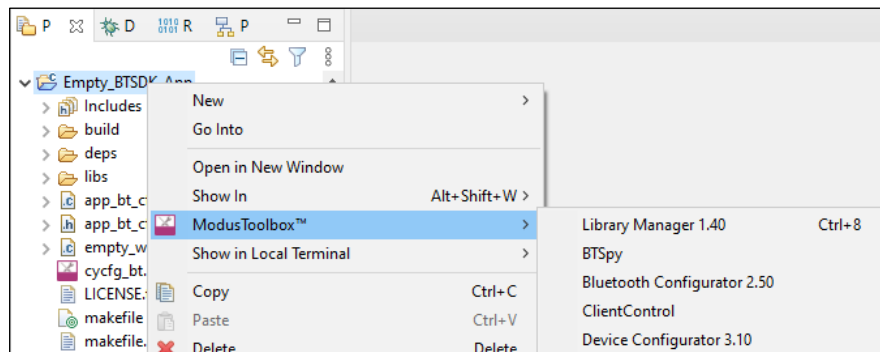
Note: *You can also launch available tools from links in the Quick Panel.*

3.3.2 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](#).

3.3.3 BTSpy and ClientControl

BTSpy is a trace utility that can be used in the AIROC™ 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.

3.3.4 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 user guide](#).

3.4 Use integrated terminal

The Eclipse IDE for ModusToolbox™ software includes an integrated terminal where you can enter various commands for the selected project. To view the terminal, click the **Terminal** tab in the bottom pane. Then, select a project in the Project Explorer to open a shell in the project directory.
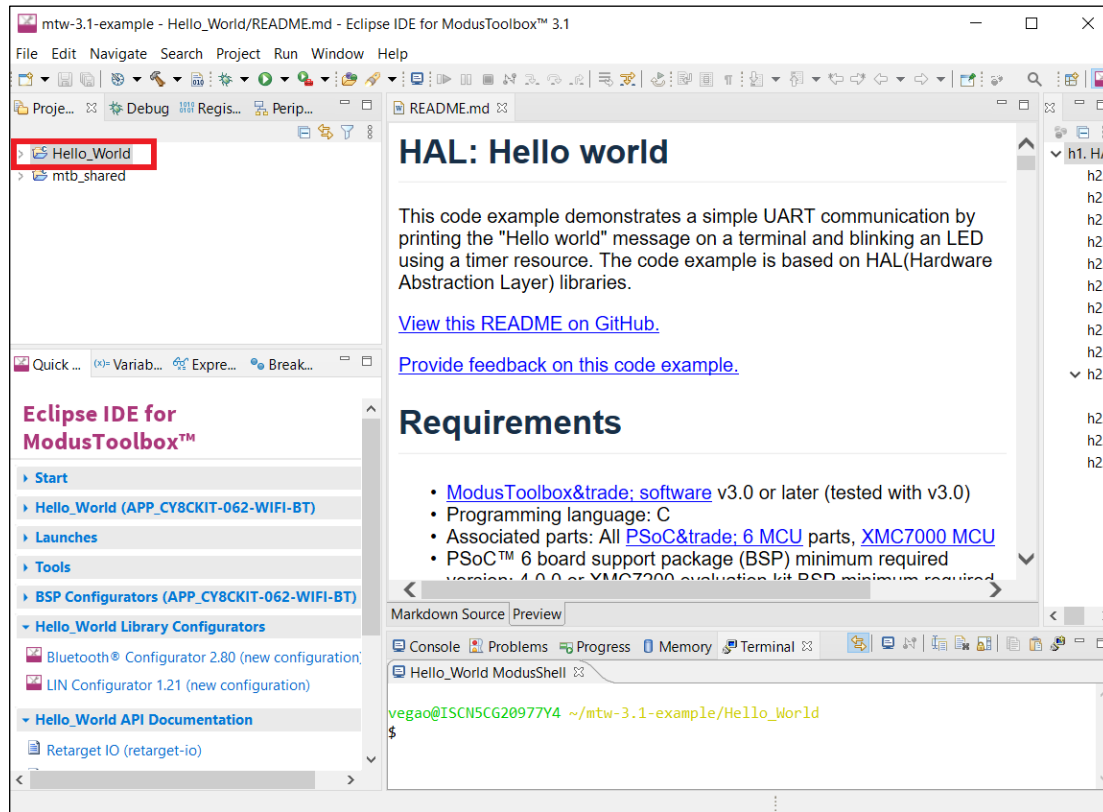
Note: *You can configure the terminal colors for the Workspace under **Window > Preferences >Terminal**.*

3.4.1 Switch projects

When you switch projects, the Eclipse IDE creates additional tabs and automatically switches between them, so that a shell in the current project directory is always in focus. To disable this automatic switching, deselect the **Track Current Project** button in the terminal toolbar.
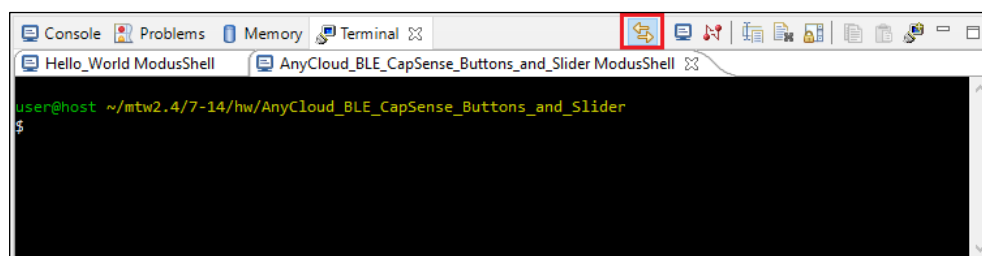
Configure applications

3.4.2   Specify alternate shell

By default, the terminal uses your login shell on Linux and macOS, or the "modus-shell" Cygwin bash shell provided with your ModusToolbox™ tools installation on Windows. You can specify a different shell from **Window > Preferences > ModusToolbox™ Tools**.
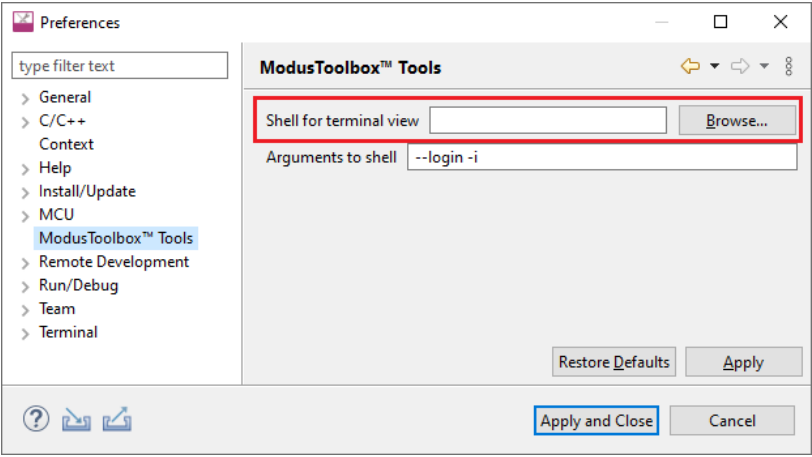
3.4.3   Update local terminal settings

By default, the local terminal uses your User home directory as the Initial Working Directory. You can specify different options from **Window > Preferences > Local Terminal**.
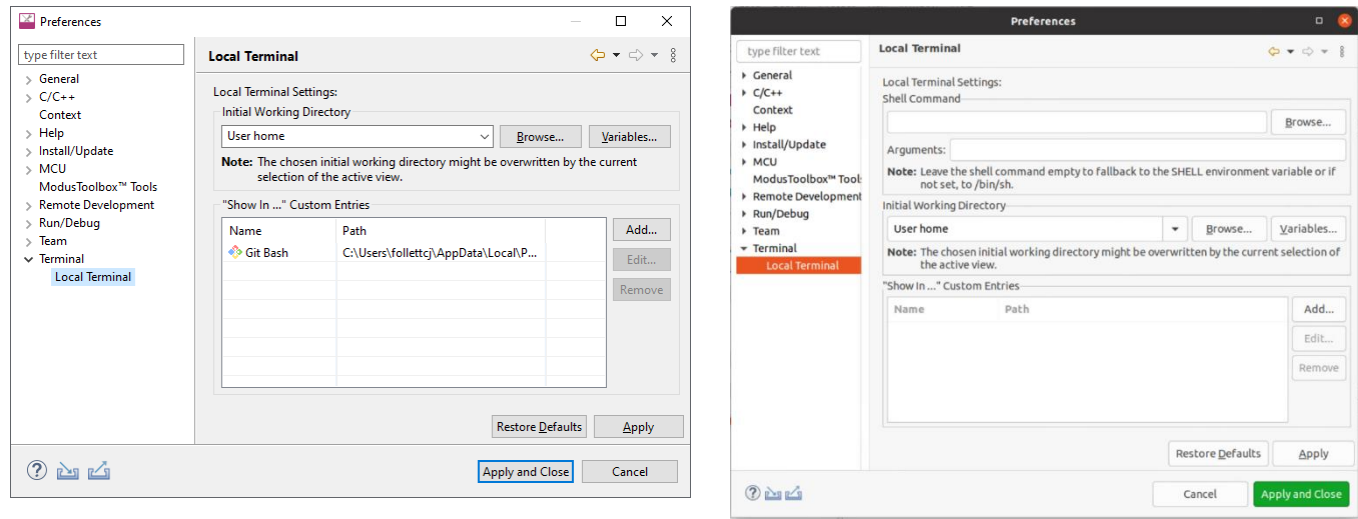
Note: Linux and macOS include additional settings for Shell Command and Arguments.

3.4.4 Connect to remote machine/board

You can also connect to a remote machine or board via serial, telnet, or SSH by clicking the **Open a Terminal** button in the terminal toolbar.
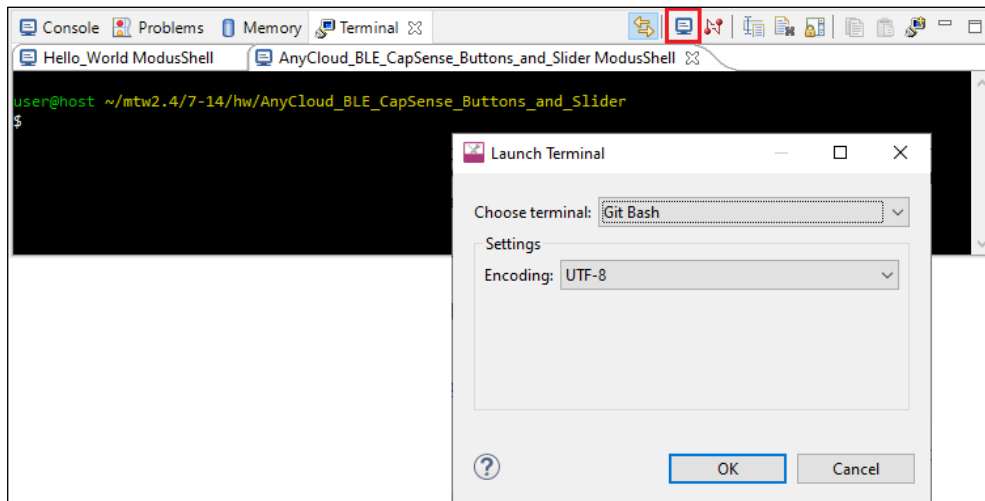
Use the **Choose terminal** pull-down menu to select the appropriate terminal.

3.5 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.
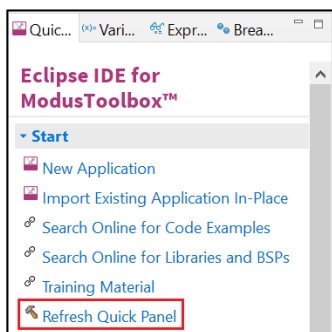
3.6 Rename application

3.6.1 Single-core application

The Eclipse IDE for ModusToolbox™ software 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 for…** 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.
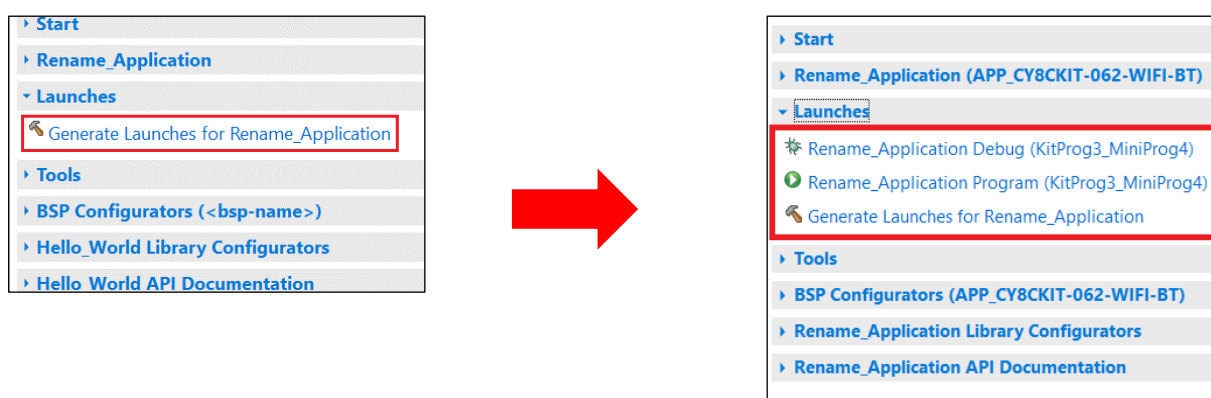
3.6.2 Multi-core application

For a multi-core application, you also use the standard Eclipse rename functionality; however, there are a few additional steps required. After renaming the main application project, you'll notice that the core subprojects become regular folders.
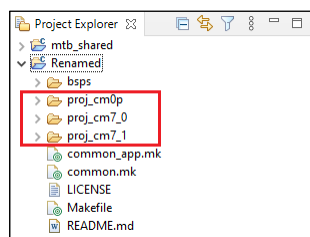
Select each folder, right-click, and select **Import as Project**.
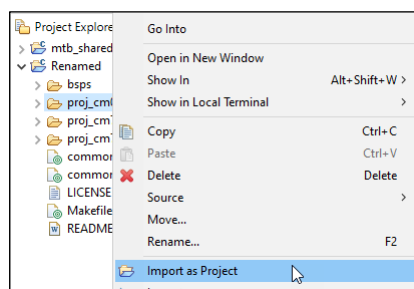
Note: *During the rename process, you may see index errors. These usually go away once the rename process is complete.*

Configure applications

After making each folder a subproject, notice that each retains the prefix of the original application name.
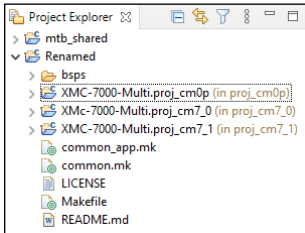
You can rename each subproject individually, but ensure they each have a unique name. You can do so by keeping the suffix.
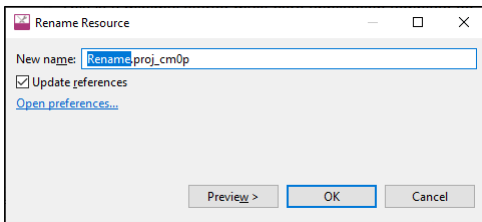
When complete with the rename process, you may have to build the application or click the **Generate Launches for …** link as described under Single-core application.

3.7 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).*

Configure applications

3. Then, select **File > Import > C/C++ > Existing Code as Makefile Project** and click **Next >**
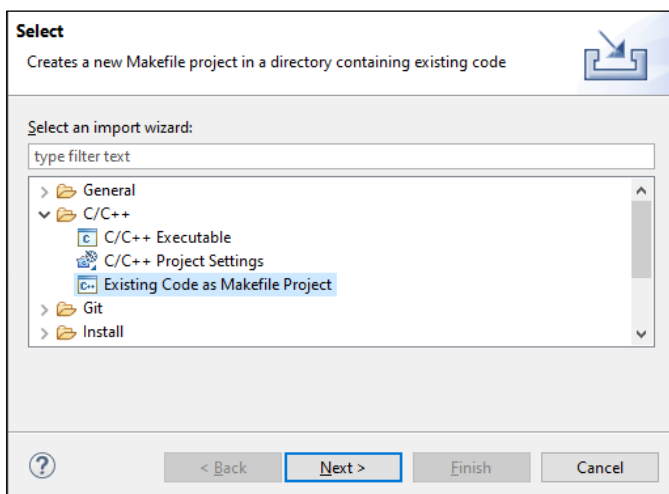
4. On the Import Existing Code page:
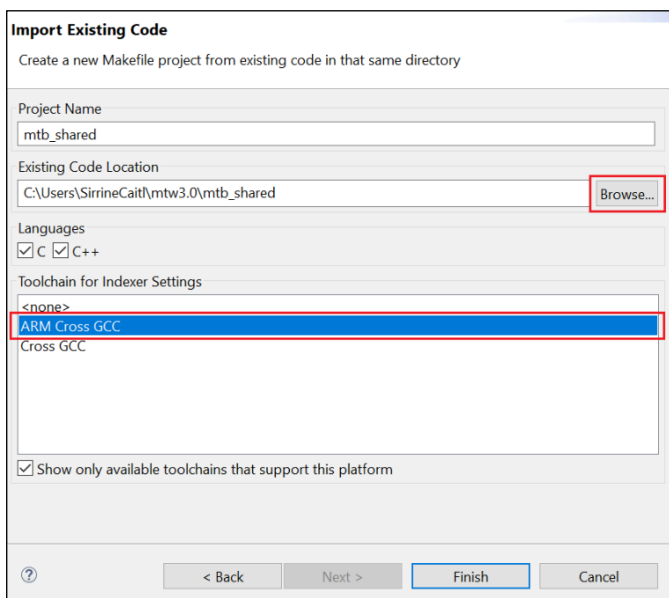
    a. Under **Existing Code Location**, click **Browse…**, navigate to the application's root directory, select the "mtb_shared" folder, and click **Select Folder**.

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

    c. Click **Finish**.

5. After the import completes, build the application.

4   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]()

4.1   Build with Make

You can build applications from the command line using Make. Refer to the [ModusToolbox™ tools package 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.

4.2   Build with IDE

After loading an application, it is best to first build everything to generate the necessary files. Click on a project in the Project Explorer. Then in the **Quick Panel**, click the "Build <app-name> Application" link.
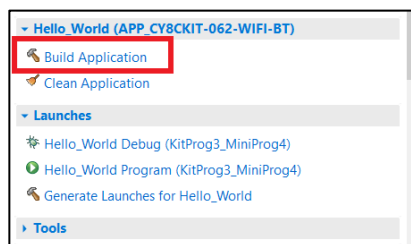
Building an application will typically allow IntelliSense to work better. It may also be useful to right-click on a project and select **Index > Rebuild** to allow IntelliSense to find references.

Messages display in the Console, indicating whether the build was successful or not.
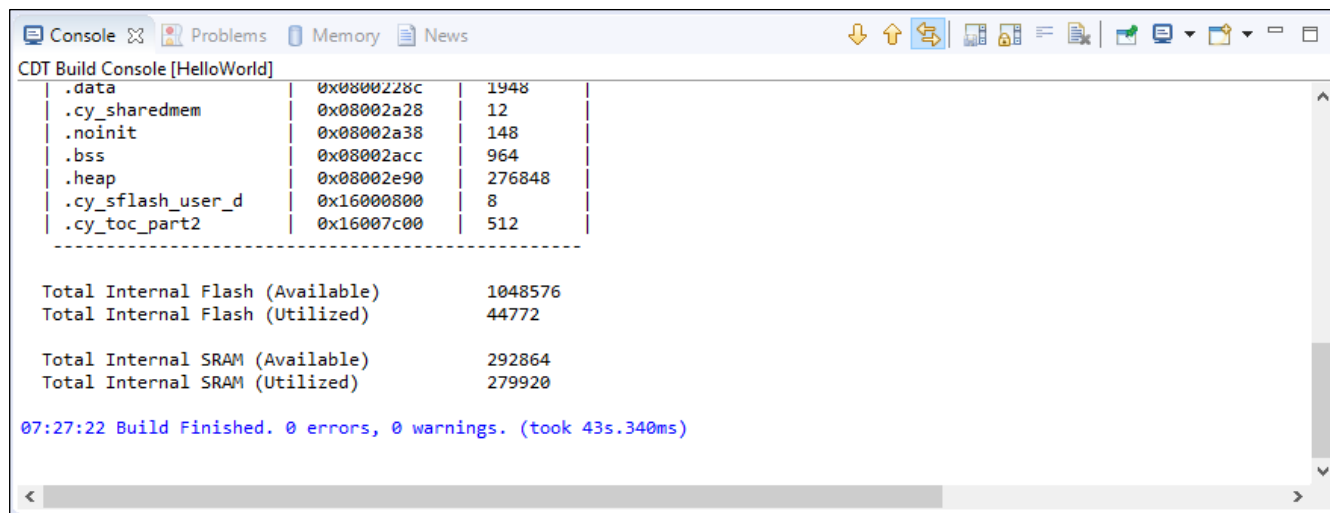
Note: Be aware that there are several Console views available.

Build applications

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.*

4.3 GCC version

ModusToolbox™ software includes GCC version 10.2 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.
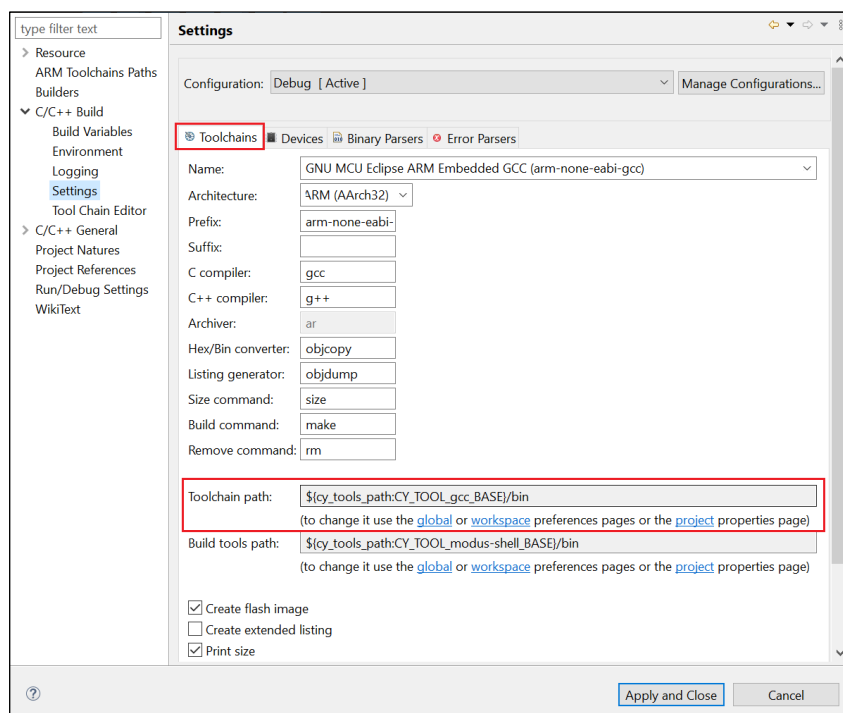
5   Program and debug

Programming and debugging are native to your chosen development environment. Infineon 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, including:

- PSoC™ MCU programming/debugging
  - Launch configurations
  - Switch to J-Link programmer/debugger
  - Changing programming interface SWD/JTAG
  - Using JTAG interface in MiniProg4
  - Erasing external memory
  - Programming eFuse
  - Debug connection options
  - Select specific CMSIS-DAP device
  - Select specific J-Link device
  - KitProg Firmware Loader
  - PSoC™ 4 flash security programming
- AIROC™ Bluetooth® programming/debugging
  - Program configuration
  - Attach configurations
  - Debug settings

5.1   PSoC™ MCU programming/debugging

5.1.1   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 Infineon 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.*

Program and debug

The Launch Configurations are shown in the Run/Debug Configurations dialog, similar to the following.
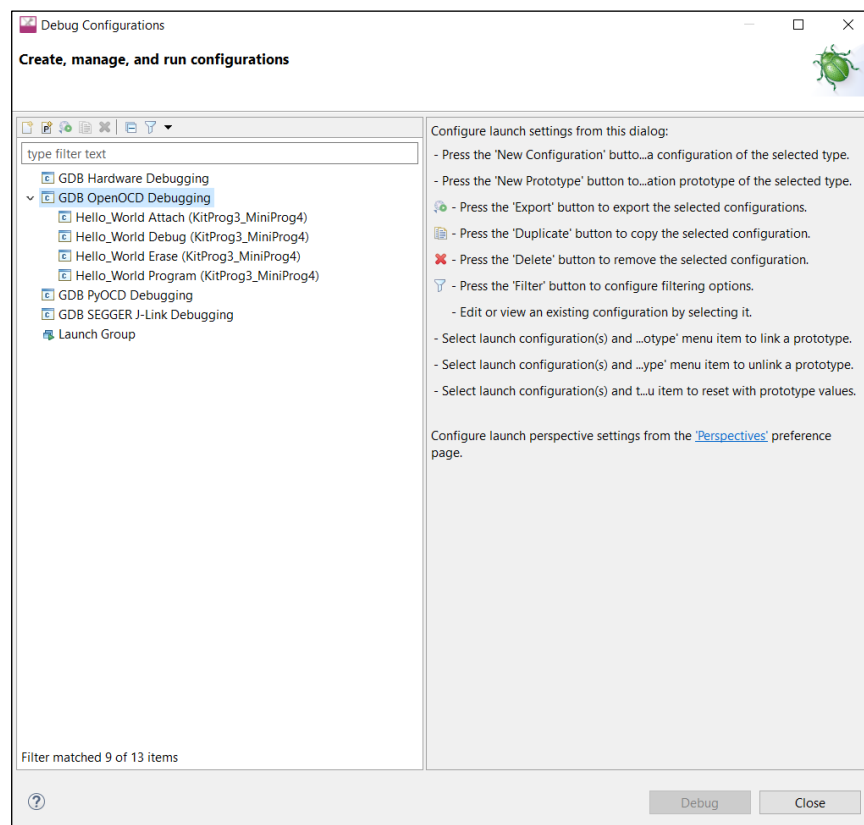
You can open these dialogs from the **Run** menu or by selecting the down arrow ▾ next to the **Run** and **Debug** commands.

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

> *Hello_World Program (KitProg3_MiniProg4)*

> *Hello_World Debug (JLink)* For more information, see [Switch to J-Link programmer/debugger](#).

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.

5.1.2 Switch to J-Link programmer/debugger

Most PSoC™-based kits use KitProg3/MiniProg4 as the default programmer/debugger, and they show those configurations in the Quick Panel:
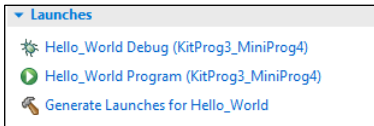
If you want to use SEGGER J-Link as the programmer/debugger, do the following to change configurations:

1. Open your ModusToolbox™ application's Makefile, and type the following variable:

```
BSP_PROGRAM_INTERFACE=JLink
```

2. If you install J-Link software in a non-default location, or if you intend to use make commands, such as `make program` and `make qprogram`, also enter the following variable:

```
MTB_JLINK_DIR=<path to J-Link>
```

Note: *If you want all applications for a BSP to use a given programmer/debugger, you can specify the* `BSP_PROGRAM_INTERFACE` *and* `MTB_JLINK_DIR` *variables in the bsp.mk file instead of the application Makefile.*

3. Save the Makefile.

4. Click **Generate Launches for <project_name>.**

When the IDE refreshes, J-Link configurations will be shown.
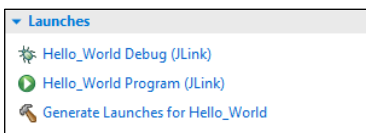
5.1.3 Using JTAG interface in MiniProg4

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:
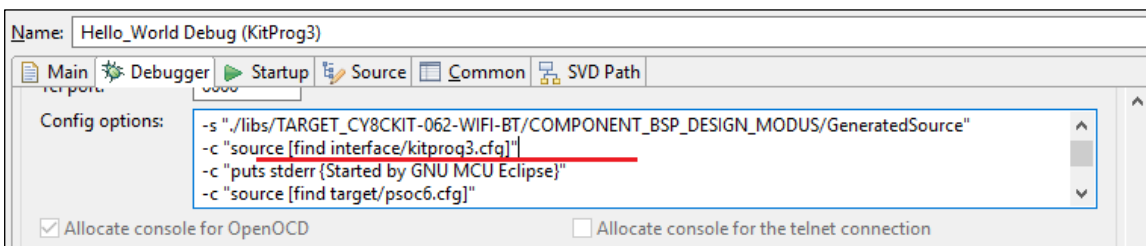
- `-c "cmsis_dap_vid_pid 0x04B4 0xF151 0x04B4 0xF152"`
- `-c "transport select jtag"`

5.1.4   Changing programming interface SWD/JTAG

Usually MCUs support programming/debugging via SWD and JTAG interfaces. The SWD interface is the default, but you can switch to JTAG, and vice-versa.

Note: There might be a difference in how the MCU functions depending on the selected debugging interface. All known issues are documented in Release Notes.

5.1.4.1   Single-core projects

Kitprog3/MiniProg4

To change the target interface for KitProg3_MiniProg4, do the following:

1. Open the Debug Configurations dialog and select the required configuration.

2. Switch to the **Debugger** tab and find the "transport select …" entry in **Config options**

3. Set to the value to either 'swd' or 'jtag' as applicable.
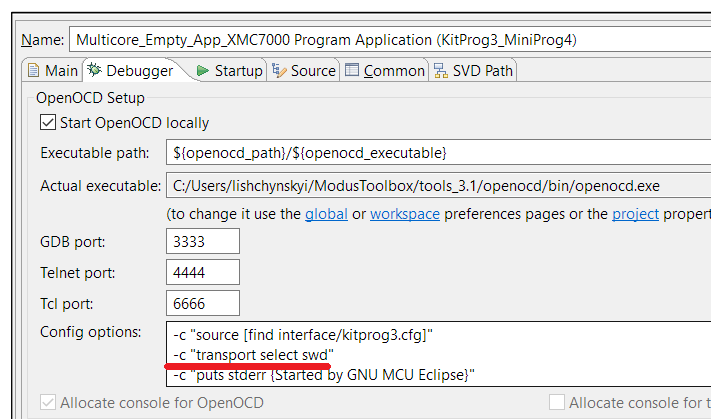
J-Link

For J-Link, there is a dedicated radio button in the Debug Configurations dialog, under the **Debugger** tab. Select the applicable option:
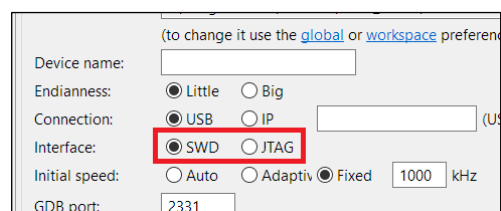
5.1.4.2   Multi-core projects

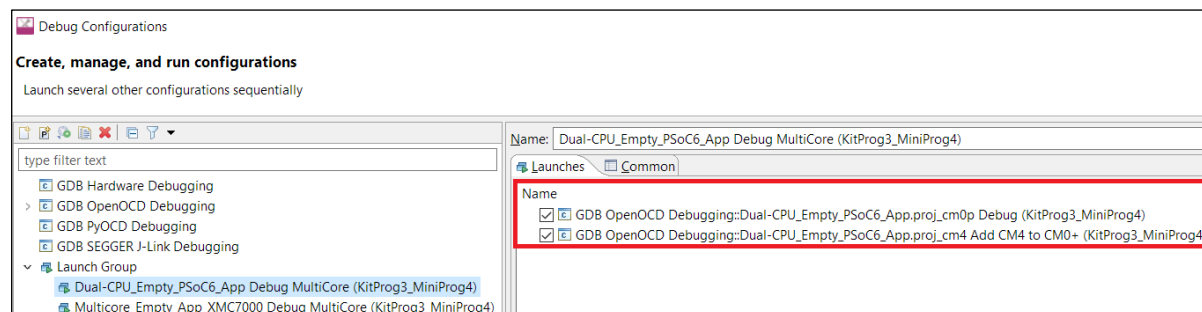There is a special case for multi-core debugging. This is usually launched via the 'Debug MultiCore' configuration. To switch from one interface to another, the same interface must be set across all involved configurations. The Debug Configurations dialog shows the list of configurations that are launched as a part of multi-core debugging.
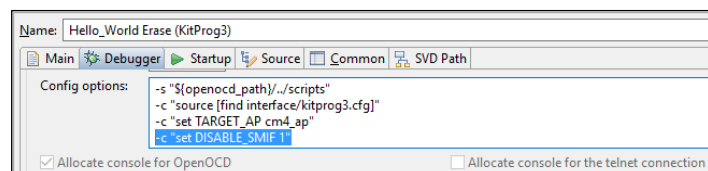

5.1.5   Erasing external memory

By default, erasing of external memory is turned off. To erase it, you must modify the "Erase" launch configuration options on the **Debugger** tab as follows:
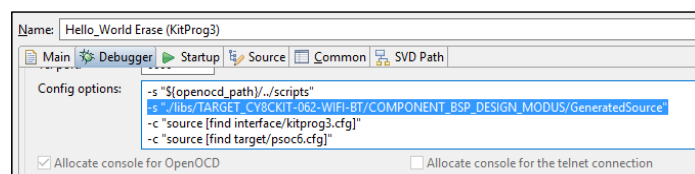
For **PSoC™ 64 Secure Boot kits**, such as CY8CKIT-064S0S2-4343W and CY8CPROTO-064S1-SB, remove the following to enable external memory erasing via SMIF:

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


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

```
-s "./bsps/<TARGET_NAME>/config/GeneratedSource"
```


5.1.6   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"
```

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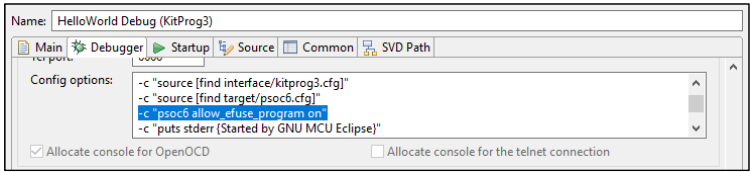
Attention: ***Since blowing an eFuse is an irreversible process, Infineon recommends programming only in mass production programming under controlled factory conditions and not prototyping stages. Programming fuses requires the associated I/O supply to be at a specific level: the device VDDIO0 (or VDDIO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

5.1.7   Debug connection options

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

| Communication Firmware	Debug Connection	More Information
Infineon-provided KitProg3	OpenOCD via CMSIS-DAP	https://www.infineon.com/cms/en/design-support/tools/programming-testing/psoc-programming-solutions/
SEGGER-provided J-Link DLL	SEGGER J-Link	SEGGER J-Link

5.1.8   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]")
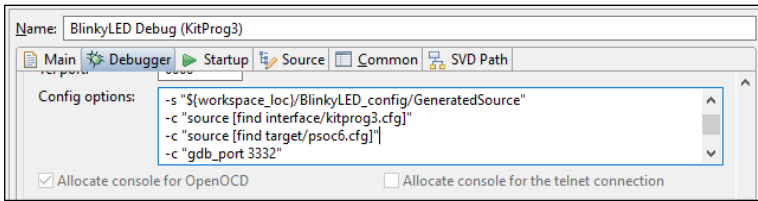
5.1.8.1   Selecting by VID and PID

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
```

Program and debug

- 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
  ```

5.1.8.2 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
  ```

5.1.8.3 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
```

5.1.9 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:
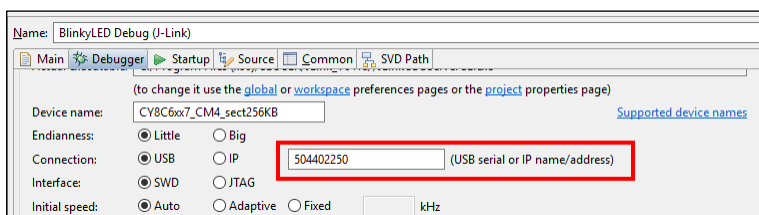
5.1.10 KitProg Firmware Loader

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 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 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.

5.1.11   Power cycle programming mode with KitProg3_MiniProg4

By default, Launch Configurations use Reset mode to program the device. However, Reset mode is not available in all situations (for example, if the XRES pin is not available on the part's package). In these cases, Launch Configurations use an alternative reset using software. However, using the software reset type is not sufficient in some cases when access to the device's DAP is restricted (such as when set by security settings).

If there is no XRES pin available and DAP access is restricted, the only way to reset a part is to use Power Cycle mode. Follow these instructions to add commands to the launch configuration and switch to Power Cycle mode.

1. Open the Launch Configuration to modify.

2. Select the **Debugger** tab.

3. In the **Config options** field, insert the following lines
    (after `-c "source [find interface/kitprog3.cfg]"`):
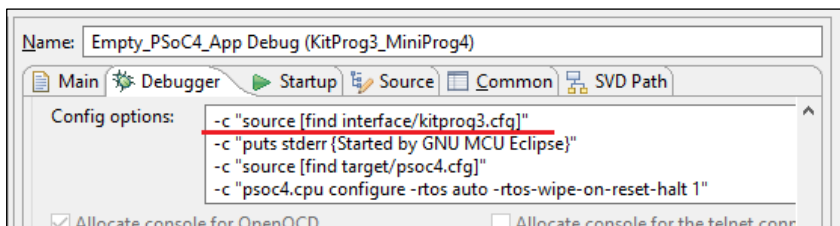
For PSoC™ 6:

```
-c "set ENABLE_POWER_SUPPLY <mV>"
-c "set ENABLE_ACQUIRE 2"
```

For PSoC™ 4:

```
-c "set ENABLE_POWER_SUPPLY <mV>"
-c "set PSOC4_USE_ACQUIRE 2"
```

Where, `<mV>` defines target voltage in millivolts, (for example, use `-c "set ENABLE_POWER_SUPPLY 3300"` for 3.3 V).

Note: *Verify the voltage range supported by the target MCU, since it can be damaged by supplying unsupported voltage. Make sure that your MCU is not powered externally before supplying power via the KitProg3_MiniProg4.*

5.1.12   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.

5.2 AIROC™ Bluetooth® programming/debugging

AIROC™ 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.
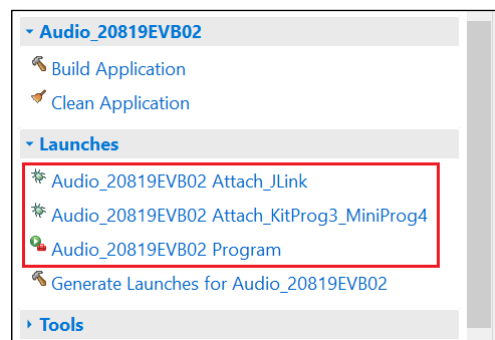
5.2.1 Program configuration

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.

5.2.2 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**

5.2.3　Debug settings

AIROC™ 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 [*AIROC™ Hardware Debugging* document](#) for more details.

Revision history

Revision	Date	Description
**	2018-11-21	New document.
*A	2019-02-22	Updates for ModusToolbox™ version 1.1.
*B	2019-10-15	Updates for ModusToolbox™ version 2.0.
*C	2019-10-29	Edits to various screen captures and related text.
*D	2020-03-20	Updates for ModusToolbox™ version 2.1.
*E	2020-04-01	Fixed broken links.
*F	2020-08-25	Updates for ModusToolbox™ version 2.2.
*G	2020-10-30	Updated Import information. Added information about exporting/sharing an application. Added information about PSoC™ 4 support. Added information about restoring shared directory.
*H	2021-03-15	Updates for ModusToolbox™ version 2.3.
*I	2021-09-22	Updates for ModusToolbox™ version 2.4.
*J	2022-09-19	Updates for ModusToolbox™ version 3.0.
*K	2022-10-18	Fixed broken links.
*L	2023-05-16	Updates for ModusToolbox™ version 3.1.

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