ModusToolbox™

Library Manager user guide

About this document

Version

1.40

Scope and purpose

This document provides information and instructions for how to use the ModusToolbox™ Library Manager.

Intended audience

Read this document to learn how to manage ModusToolbox™ BSPs and libraries for your application.

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™ user guide
- Eclipse IDE for ModusToolbox™ user guide
- Project Creator user guide

Table of contents

1 Overview 3

1.1 Version 3
1.2 Manifests 3
1.3 MTB flow 3
1.4 LIB flow 6

2 Launch the Library Manager 7

2.1 make command 7
2.2 Eclipse IDE 7
2.3 Stand-alone GUI mode 7
2.4 Offline mode 8
2.5 Non-GUI command line interface (CLI) 8

3 Working with BSPs/libraries (MTB flow) 9

3.1 Update indirect dependency libraries 10
3.2 Add/remove BSPs 10
3.3 Select Active BSP 11
3.4 Share/unshare BSPs and libraries 11
3.5 Change BSP/library version 11

4 Working with BSPs/libraries (LIB flow) 13

4.1 Adding BSPs/libraries 13
4.2 Remove BSPs/libraries 14
4.3 Update shared BSPs/libraries 14
4.4 Change BSP/library version 14

5 GUI description 15

5.1 Menus 15
5.2 Fields 15
5.3 Tabs 16
5.4 Buttons 16
5.5 Message console 17
5.6 Status indicator 17

6 Manual library management 18

6.1 Acquiring a library (MTB flow) 18
6.2 Acquiring dependencies (MTB flow) 18
6.3 Acquiring a library (LIB flow) 19
6.4 Acquiring dependencies (LIB flow) 20

7 Tool change description 21

1 Overview

The Library Manager provides a GUI to select which Board Support Package (BSP) should be used when building a ModusToolbox™ application. The tool collects a list of available and currently selected BSPs and libraries, as well as all the necessary metadata from a webservice. The tool allows you to add and remove BSPs and libraries, as well as change their versions.

In ModusToolbox™ 2.2 and later, we structured applications with the MTB flow and *.mtb* files. Using this flow, applications can share BSPs and libraries. If needed, different applications can use different versions of the same BSP/library. Sharing resources reduces the number of files on your computer and speeds up subsequent application creation time. Shared BSPs, libraries, and versions are located in a "*mtb_shared*" directory adjacent to your application directories. You can easily switch a shared BSP or library to become local to a specific application, or back to being shared.

Most example applications use the MTB flow. However, there are still older applications that use the previous flow, now called the LIB flow (*.lib* files), and these applications generally do not share BSPs and libraries. ModusToolbox™ fully supports both flows, but it only supports one flow or the other for a given application. If an application uses the MTB flow, the system will process *.mtb* files only and ignore any *.lib* files.

1.1 Version

All BSPs and libraries have a version. Versions apply to both the MTB and LIB flows. See also BSP and library versions later in this document.

1.1.1 Dynamic

If a BSP or library uses a "Latest" tag, then it is dynamic. ModusToolbox™ will download and use the appropriate version specified in the manifest and attempt to resolve any potential conflicts with different library versions. Using a "Latest" tag means the library will update to the latest version whenever you run the `make getlibs` command, including when you click the Library Manager **Update** button.

1.1.2 Fixed

By selecting a specific version from the pull-down menu in the Library Manager GUI, you assign a fixed version to the BSP or Library that does not change, unless you manually change it.

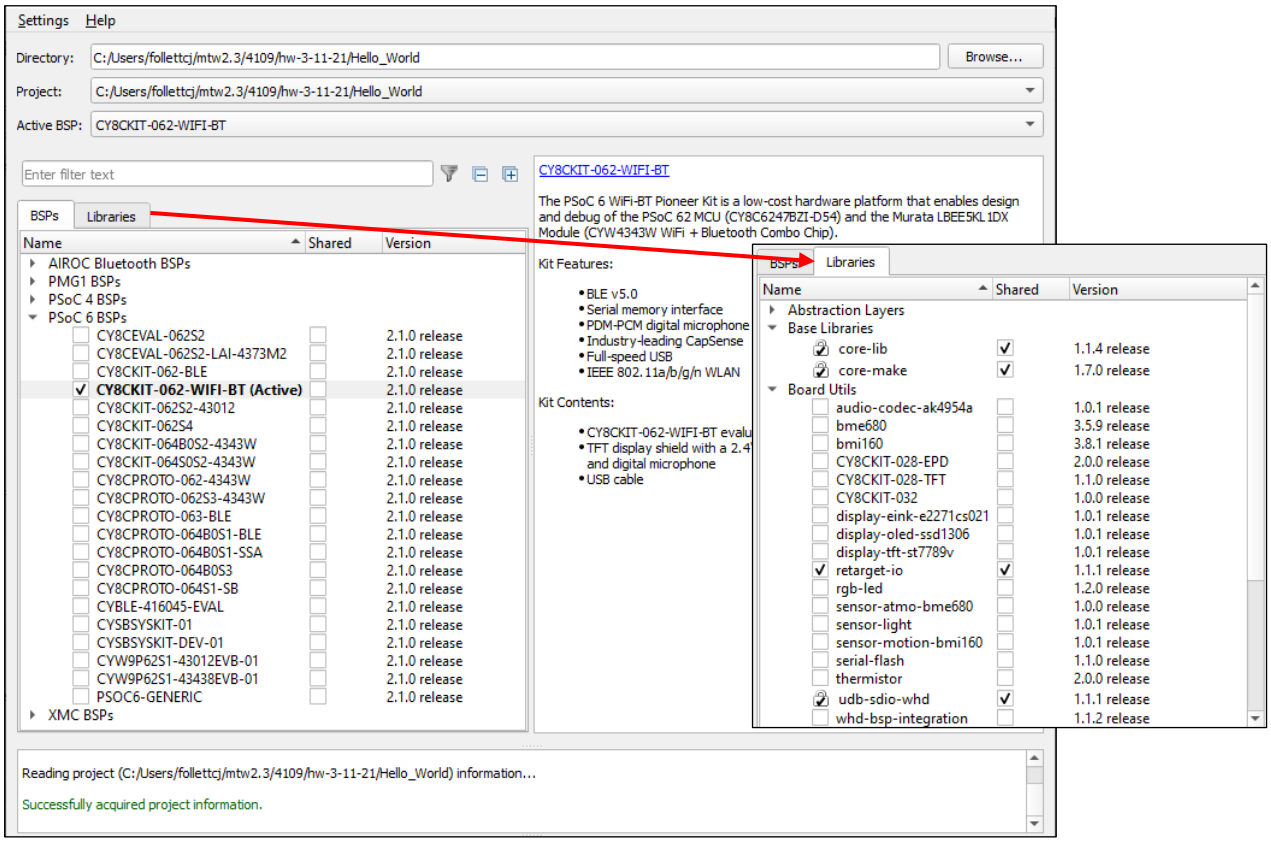
1.2 Manifests

Manifests are XML files that tell the Library Manager how to discover the list of available BSPs, libraries, and library dependencies. Manifests apply to both the MTB and LIB flows. For more information, refer to the Manifest chapter in the ModusToolbox™ user guide.

1.3 MTB flow

The MTB flow applies to ModusToolbox™ 2.2 and later releases. This flow uses *.mtb* files to manage libraries. The Library Manager runs the `make getlibs` command to process the *.mtb* files, pull the libraries from the specified git repos, and store them in the specified location. The system also creates a file called *mtb.mk* in the application's *libs* subdirectory. The build system uses that file to find all the libraries required by the application.

The following shows the Library Manager for a typical PSoC 6 BSP and the MTB flow, with **BSPs** and **Libraries** tabs. The key items are the **Shared** column and the lock symbols for libraries that are included because they are dependencies of other libraries. See Working with BSPs/libraries (MTB flow) for more details about using the tool in the MTB flow.



The following concepts are key to understanding the MTB flow:

1.3.1 .mtb file

This file provides information about the associated BSP/library. It contains:

- A URL to a git repository somewhere that is accessible by your computer such as GitHub.
- A git Commit Hash or Tag that tells which version of the library that you want.
- A path to where the library should be stored.

A typical *.mtb* file looks like this:

```
https://github.com/Infineon/TARGET_CY8CKIT-062S2-43012/#latest-
v1.X#$$ASSET_REPO$$/TARGET_CY8CKIT-062S2-43012/latest-v1.X
```

The variable `$$ASSET_REPO$$` points to the root of the shared location. If you want a library to be local to the application instead of shared, use `$$LOCAL$$` instead of `$$ASSET_REPO$$`.

1.3.2 Direct dependency

This is a library or BSP that is directly referenced by an application. In many applications, the only direct dependency is a BSP. In other applications, there will be additional direct dependencies such as Wi-Fi or Bluetooth libraries. You may also have changed the location or version of an indirect library, which makes that library a direct dependency.

You manage the *.mtb* files for direct dependencies, and *.mtb* files should be checked into source control along with your application source code. A *.mtb* file for a direct dependency is typically stored in the application's *deps* subdirectory by default. You can store it anywhere inside the application **except** in the application's *libs* subdirectory.

1.3.3 Indirect dependency

This is a library that was included indirectly as a dependency of a BSP or another library. The system stores the code in the appropriate directory and chooses the appropriate version specified in the manifest, as well as resolves any potential conflicts with different library versions.

The system finds indirect dependencies for each library using information that is stored in a manifest file. For each indirect dependency found, the Library Manager places an *.mtb* file in the application's *libs* subdirectory. The tool also creates a locking_commit.log file in the application's *deps* subdirectory to record the locked release version of those libraries.

You do not manage *.mtb* files for indirect dependencies, and they do not need to be checked into source control. The *.mtb* file for an indirect dependency is stored in the application's *libs* subdirectory and must **not** be moved.

1.3.4 Shared

A shared BSP/library includes the code for the associated *.mtb* file, and the BSP/library can be shared by multiple applications in the same directory structure or workspace. When creating a new application, BSPs/libraries can be shared and placed in an *mtb_shared* directory adjacent to the application directory(ies), based on the "default_location" in the manifest file. When you add a BSP/library and specify it as shared, it is also included in the *mtb_shared* directory. These BSPs/libraries do not need to be checked into source control. All the code can be re-downloaded at any time from the information in the *.mtb* file.

You can change the name and location of the shared directory using make variables `CY_GETLIBS_SHARED_NAME` and `CY_GETLIBS_SHARED_PATH`. For more details about the ModusToolbox™ build system and make variables, refer to the ModusToolbox™ user guide.

1.3.5 Local

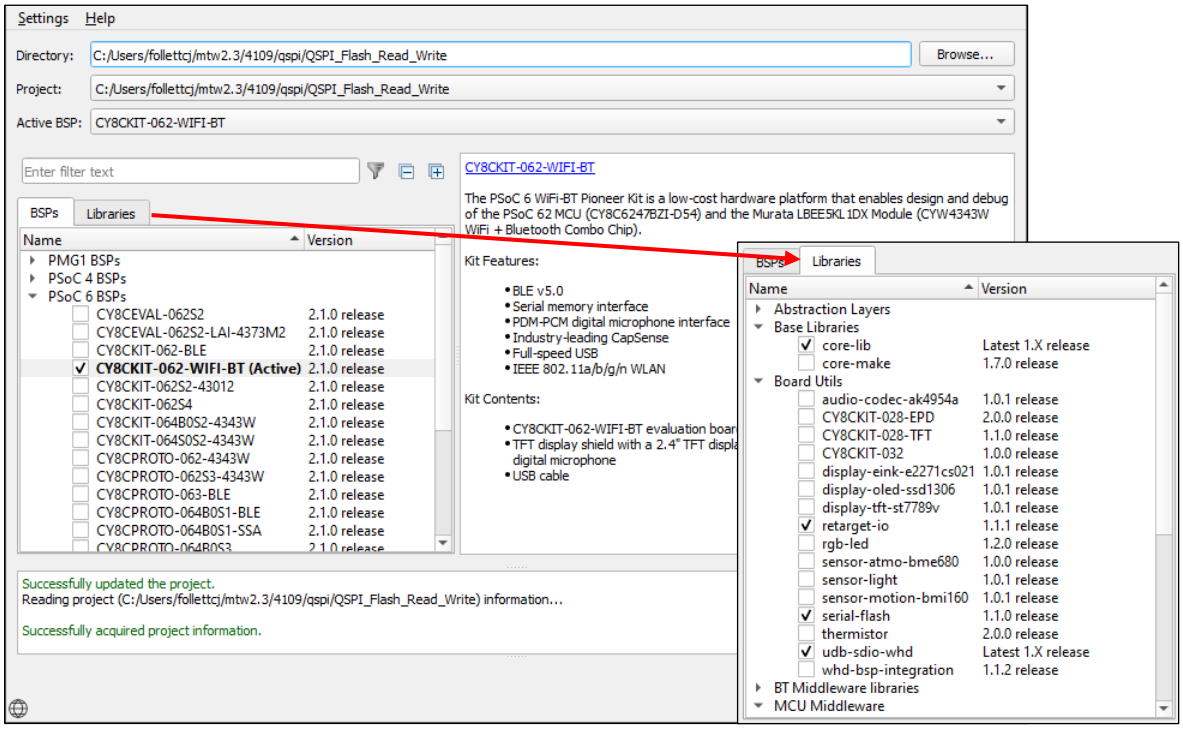
A local BSP/library includes the code for the associated *.mtb* file, and the BSP/library is used only for a specific application. The local BSP/library is stored in the *libs* subdirectory of the specific application. These BSPs/libraries do not need to be checked into source control. All the code can be re-downloaded at any time from the information in the *.mtb* file.

If you change BSPs and libraries to be local to an application (see Share/unshare BSPs and libraries later in this guide), then the source code for these are located in the *libs* subdirectory inside the application directory.

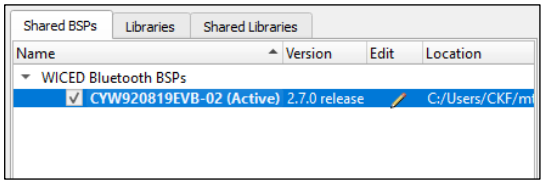
1.4 LIB flow

Some older applications before ModusToolbox™ 2.3 use the LIB flow, where libraries are managed with *.lib* files.

In the Library Manager, these applications also have the **BSPs** and **Libraries** tabs like the MTB flow, but they do not have a **Shared** column and they do not distinguish between libraries that are included by the application vs. libraries that are included because other libraries depend on them. See Working with BSPs/libraries (LIB flow) for more details.



Some applications also have **Shared BSPs** and **Shared Libraries** tabs.



These are the concepts for the LIB flow.

- **Local lib file –** This is a .lib file present in the project whose library is being edited.
- **Shared lib file –** This is a .lib file that is used indirectly via the shared library mechanism.
- **Target project –** This is the project directory whose library is being edited.
- **Fully editable mode –** You can add, remove, and change the version of the library.
- **Restricted edit mode –** You can manage the version of the selected library, but you cannot add or remove the library.

2 Launch the Library Manager

There are numerous ways to launch the Library Manager, and that depends on how you use the various tools included with ModusToolbox™ software.

2.1 make command

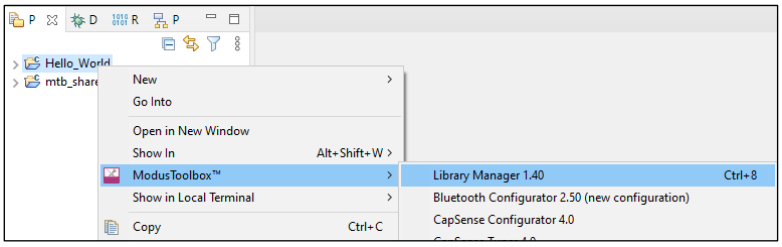
As described in the ModusToolbox™ user guide build system chapter, you can run numerous `make` commands in the application directory, such as launching the Library Manager. After you have created a ModusToolbox™ application, navigate to the application directory and type the following command in the appropriate bash terminal window:

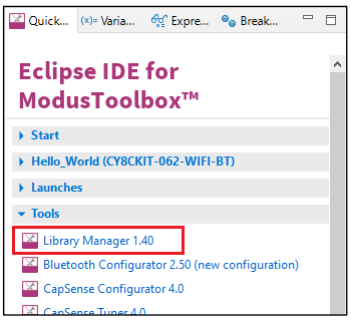
```
make modlibs
```

This command opens the Library Manager GUI for the specific application in which you are working.

2.2 Eclipse IDE

If you use the Eclipse IDE for ModusToolbox™, you can launch the Library Manager for the selected application. In the Project Explorer, right-click on the project and select **ModusToolbox™ > Library Manager <version>**. You can also click the Library Manager link in the IDE Quick Panel.





Similar to the `make` command method, launching the Library Manager using the Eclipse IDE opens the tool for the selected application. Refer to the Eclipse IDE for ModusToolbox™ user guide for details about the IDE.

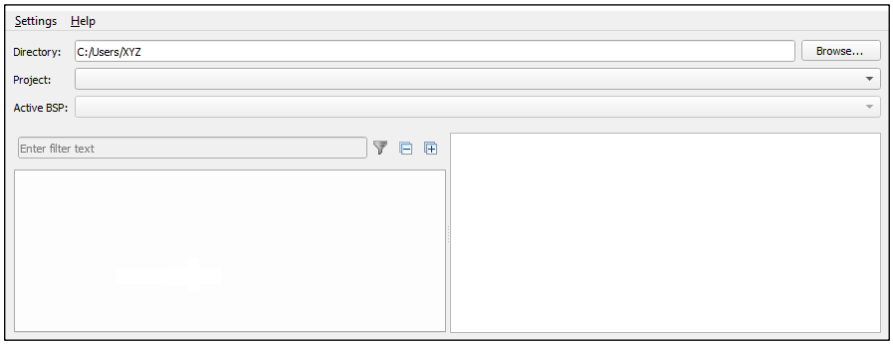
2.3 Stand-alone GUI mode

You can also launch the Library Manager in stand-alone mode by running its executable as appropriate for your operating system (for example, double-click it or select it using the Windows **Start** menu). By default, it is installed here:

<install_dir>/ModusToolbox/tools_<version>/library-manager-<version>

When run in stand-alone mode, the Library Manager opens with the target directory set as <user-home> or as the directory selected from a previous stand-alone session.

If you haven't opened the Library Manager tool previously, it starts at your home directory and looks for directories with a Makefile, and then it opens the first application it finds in that directory. If the tool does not find an application, it doesn't display any BSPs or libraries.



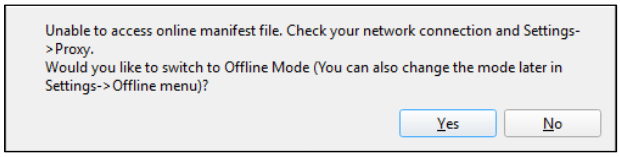
Click the **Browse…** button next to the **Directory** field and navigate to the directory that contains one or more applications. Then, select from the **Project** pull-down menu as needed.

Note: *The next time you open the Library Manager in stand-alone mode, it will open with the most recently selected application.*

2.4 Offline mode

If the Library Manager cannot connect to the internet when launching, it displays a message in the console that it cannot access the manifest file. Adjust your internet and/or proxy settings (from the **Settings** menu), and then click the **Retry** button to re-read library information.

If you have the ModusToolbox™ Offline Content bundle installed and run the Library Manager without an internet connection, the following dialog displays:



If you click **Yes**, the Library Manager switches to offline mode and reads library information from the offline content bundle. If you click **No**, you can adjust internet and/or proxy settings, and then click **Retry** to re-read library information. Refer to the ModusToolbox™ user guide for more details about the offline content bundle.

2.5 Non-GUI command line interface (CLI)

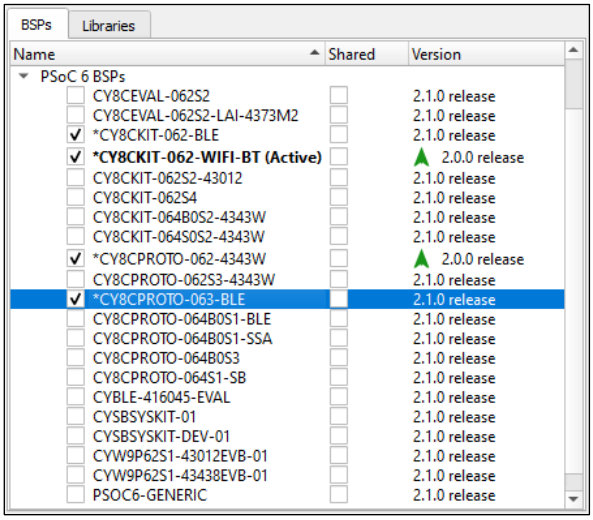
You can also open a non-graphical interface from the command line. However, there are only a few reasons to do this in practice. The primary use case would be part of an overall build script for the entire application. For information about command line options, run the `library-manager-cli` executable using the `-h` option.

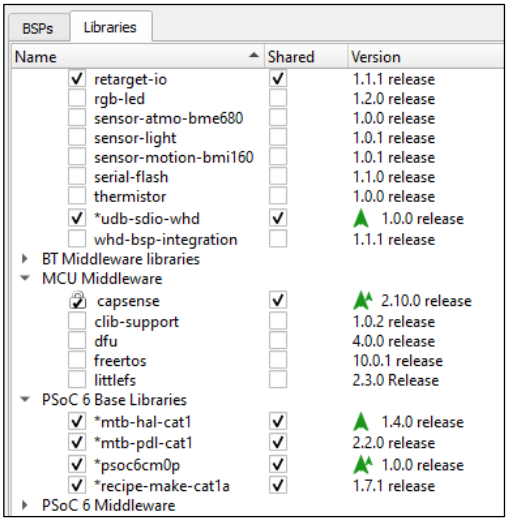
3 Working with BSPs/libraries (MTB flow)

This section covers various ways to update BSPs and libraries for the MTB flow, including:

- Update indirect dependency libraries
- Add/remove BSPs or libraries
- Select Active BSP
- Share/unshare BSPs and libraries
- Change BSP/library version

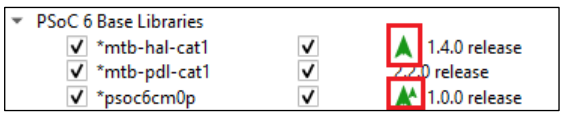
When using the MTB flow, the **BSPs** and **Libraries** tabs show selection check boxes to the left of each of the listed BSPs and libraries. Selected check boxes represent BSPs/libraries included in the application, while those without selected check boxes are not included. Check boxes under the **Shared** column indicate if a BSP/library is shared or local, and each BSP/library shows a version under the **Version** column.





When you make a change to a BSP or library, an asterisk (*) displays next to it indicating a change is pending, but it has not yet been updated. Changes will take effect when you click the **Update** button.

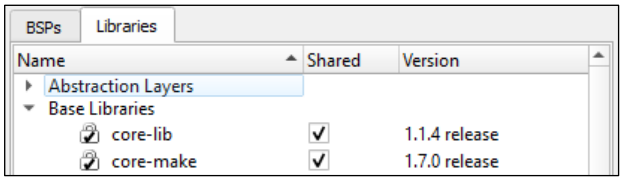
When a BSP/library has a newer version available than the one currently selected, a “newer version” symbol displays next to the version number, indicating that a newer major or minor version is available.



If you hover the mouse cursor over the symbol, a tooltip displays the type of version.

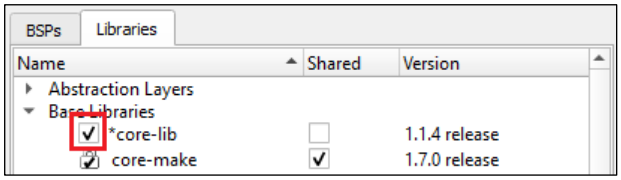
3.1 Update indirect dependency libraries

Several libraries included in your application show a lock symbol by default. This symbol indicates that a library is an indirect dependency, and it is required by the BSP and/or another library.

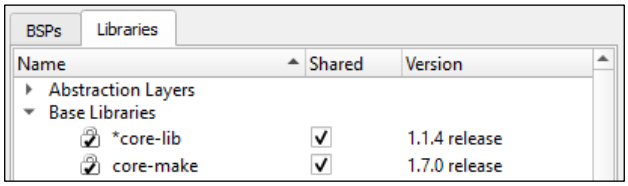


These libraries cannot be removed using the Library Manager; however, you can change whether it is shared or not, and you can select a different version.

When you make a change to an Indirect Dependency library, the lock symbol changes to regular check box. This indicates that after you click **Update**, the library is now a direct dependency and it is your responsibility to manage it either as a local library and/or with the selected version.

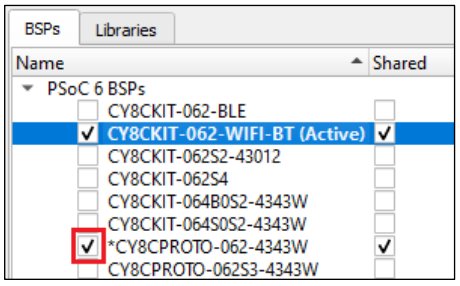


To change the library back to an Indirect Dependency, click on the check box. The icon reverts back to the lock symbol, and the **Shared** and **Version** settings revert back to the defaults. You must click the **Update** button to process the changes.



3.2 Add/remove BSPs

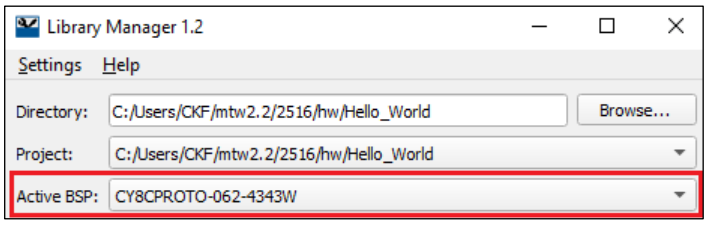
Add BSPs to your application and remove them by selecting and deselecting the check boxes to the left of the BSPs. Click the **Update** button to process the change(s). The message console displays the progress and indicates when changes are complete.



Note: *Use caution when removing all Cypress BSPs from your application. BSPs are responsible for ensuring key libraries are present, including "psoc6make" for PSoC 6 projects and "baselib" for Bluetooth projects. Before you remove all Cypress BSPs, make sure a custom BSP is present in your application; otherwise, the Library Manager tool may not function correctly. For information on creating and customizing BSPs, refer to the ModusToolbox™ user guide.*

3.3 Select Active BSP

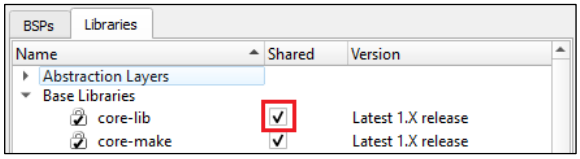
The **Active BSP** for the application is shown in bold text with "(Active)" next to it. An application can only have one active BSP for which all settings apply. If your application has more than one BSP included (shown with selected check boxes), you can use the **Active BSP** pull-down menu to select one of the other BSPs to be active.



When you click the **Update** button, the system displays progress in the message console and updates your application's makefile `"TARGET="` field to specify the new BSP as active.

3.4 Share/unshare BSPs and libraries

When you create applications, BSPs and libraries can be shared or local depending on the application and manifest. Shared BSPs/libraries are included in a *mtb_shared* directory, which is adjacent to your application directory. BSPs/Libraries that are local (aka, **not** shared) are included in the *libs* subdirectory inside your application's directory.

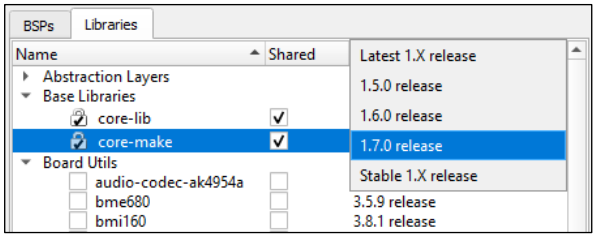


If you deselect the **Shared** check box, a new BSP/library will be created in the *libs* subdirectory, and it will **not** be removed from the *mtb_shared* library. This is because the system assumes a shared BSP/library is being used by another application. However, if you select the **Shared** check box for a BSP/library already included in the application and click **Update**, that BSP/library will be moved from the *libs* subdirectory to the *mtb_shared* directory.

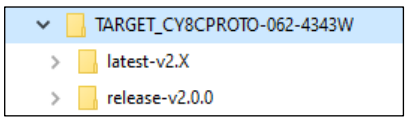
Click the **Update** button to process the change(s). The message console displays the progress and indicates when changes are complete.

3.5 Change BSP/library version

Libraries have a specific version, by default. You can change a BSP/library version by selecting the pull-down menu under the **Version** column and choosing another version. See BSP and library versions for a detailed description.



When you change versions for a shared BSP/library, the tool creates a new subdirectory for the added version to ensure that the existing version is available for other applications. For example:



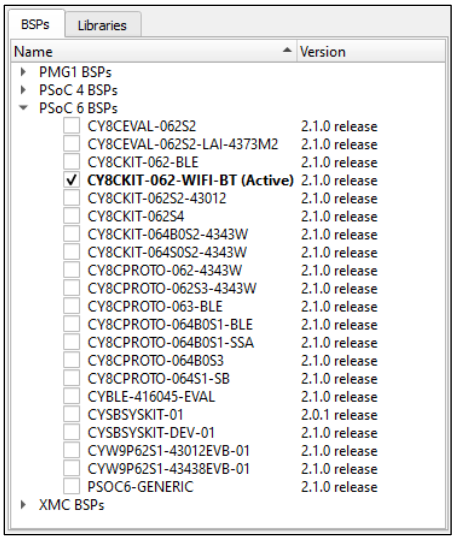
However, when you change a version for a local BSP/library, the system will **not** create a new local subdirectory. Instead, the version of the BSP/library will just be updated with the appropriate tag.

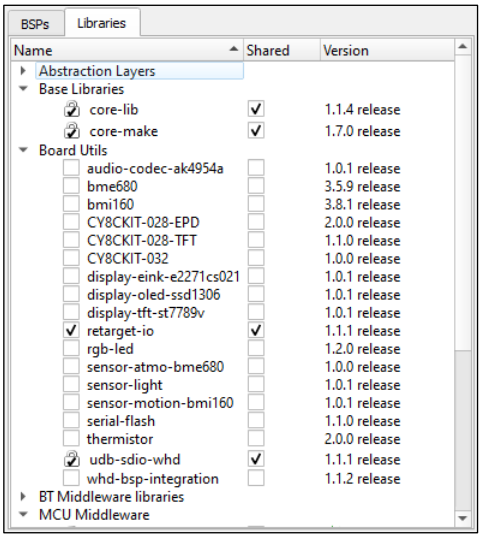
4 Working with BSPs/libraries (LIB flow)

This section covers various ways to update BSPs and libraries for the LIB flow, including:

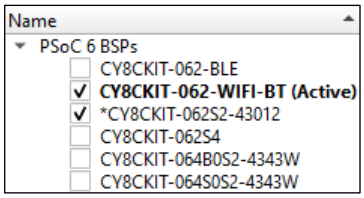
- Add BSPs/Libraries
- Remove BSPs/Libraries
- Update Shared BSPs/Libraries
- Change BSP/Library Version

When using the LIB flow, the **BSP** and **Libraries** tabs show selection check boxes next to each of the listed BSPs and libraries supported in the application. Selected check boxes represent BSPs/libraries already included in the application, while those without selected check boxes are not included. The Active BSP for the application is shown in bold text with "(Active)" next to it.



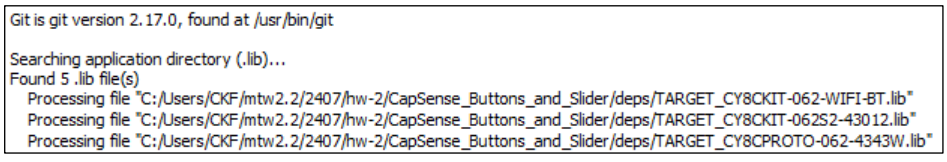


Note: *When you make a change to a BSP or library, an asterisk (*) displays next to it indicating a change is pending, but it has not yet been updated. Changes will take effect when you click the **Update** button.*



4.1 Adding BSPs/libraries

To add a BSP/library to your application, select the check box for one or more item on each tab and click **Update**. The tool starts updating the project and displaying progress in the status box.



When complete, the additional items will display with selected check boxes.

4.2 Remove BSPs/libraries

To remove a BSP/library from your application, deselect the check box for one or more item on each tab, and click **Update**.

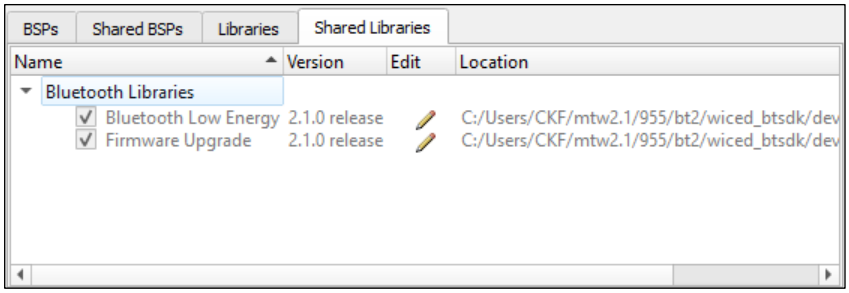
Note: *Use caution when removing all Cypress BSPs from your application. BSPs are responsible for ensuring key libraries are present, including "psoc6make" for PSoC 6 projects and "baselib" for Bluetooth projects. Before you remove all Cypress BSPs, make sure a custom BSP is present in your application; otherwise, the Library Manager tool may not function correctly. For information on creating and customizing BSPs, refer to the ModusToolbox™ user guide.*

Click **Update** and the tool starts updating the project and displaying progress in the status box.

When complete, the removed items will display in the Library Manager without selected check boxes. On disk, the libraries are removed by creating a special .lib file with a commit set to a sentinel value that indicates the library is removed.

4.3 Update shared BSPs/libraries

For certain applications, BSPs and Libraries are shared. In these cases, you may see additional tabs: **Shared BSPs** and **Shared Libraries**.



You cannot update items on these shared tabs directly. Instead, you must click the **Edit** icon . This opens another instance of the Library Manager where the BSP and Libraries are not shared. Edit the BSP/Library as described previously, and then close the second instance of the Library Manager.

Note: *Even though the BSP/Library was updated in the 2nd instance of the Library Manager, the changes will not be displayed in the shared tabs until you close and reopen the 1st instance.*

4.4 Change BSP/library version

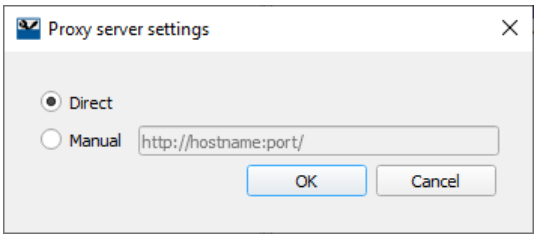
This is the same process as updating versions for the MTB flow.

5 GUI description

5.1 Menus

The Library Manager has two menus, as follows:

- **Settings** –
  - **Offline**: Check box to toggle between online and offline mode and read the local copy of the manifest file installed from the offline content bundle. See offline mode for more details.
  - **Proxy Settings**: Opens a dialog to specify direct or manual proxy settings.



- **Help** –
  - **View Help**: Opens this document.
  - **About**: Displays tool version information.

5.2 Fields

The Library Manager contains the following fields at the top of the GUI:

- **Directory** – Location of the application's top-level directory, which contains one or more ModusToolbox™ makefile projects. Use the **Browse…** button to select a different directory, if needed.
- **Project** – Location of the application's selected makefile. Use the pull-down menu to select another project, if applicable.
- **Active BSP** – The name of the currently active target BSP. Use the pull-down menu to choose any of the boards with a selected check box in the **BSPs** tab. You can have multiple sets of BSP code in a project; however, the make project is built only for the selected **Active BSP**.
- **Filter text** – Field used to show only the BSPs/libraries that match the text entered.
- **Show/Hide** – Toggle to show only the BSPs/libraries with check boxes selected or show all BSPs/libraries. If there is filter text, then only BSPs/libraries with check boxes selected and BSPs/libraries with matching text will be shown instead of all.
- **Collapse All** – Click to collapse all item trees.
- **Expand All** – Click to expand all item trees.

5.3 Tabs

The Library Manager may display one or more of the following tabs, depending several variables:

- **BSPs tab** – Shows available BSPs and versions. This tab is hidden if there are no BSPs in the project and the project type is a library project or if the project uses shared BSPs.
- **Libraries tab** – Shows libraries supported by the Active BSP. This tab is hidden if the project type is a library project and if the project contains zero local libraries.
- **Shared BSPs tab (LIB Flow)** – Shows a read-only view of BSPs that are available via the LIB flow shared library mechanism. This tab is hidden if the project is not using any shared BSPs or if the project type is a library project.
- **Shared Libraries tab (LIB Flow)** – Shows a read-only view of LIB flow code libraries that will be used by the project. This tab is hidden if the project is not using any shared libraries or if the project type is a library project.

5.3.1 Shared check box (MTB flow)

This column contains a check box that indicates if the associated BSP/Library is shared or not.

5.3.2 BSP and library versions

The Library Manager contains a **Version** column for each BSP and library shown in the tool. You can select "Latest X.Y release" or "X.Y.Z release." These represent tags for versions of BSPs/libraries in Cypress GitHub repos:

- The "latest-vX.Y" tag is updated whenever you run `make getlibs` (such as by clicking the **Update** button) to point to a newer, backward-compatible version of the library when Cypress releases it.
- The "release-vX.Y.Z" tag always points to a specific, official release, and it does not change.

When you create a new application from a Cypress code example, the application converts BSPs and libraries to use the "X.Y.Z release" version, by default. This ensures that they will not be updated automatically, unless you change the version to "Latest X.Y release."

When you open the Library Manager for that application and make any kind of change, the change summary displays in the console. If a "latest-vX.Y" tag has a newer version, the console displays a warning about the "latest-vX.Y" tag. It includes a hyperlink to open a dialog that explains how the "latest-vX.Y" tag auto-update works.

If you click **Update**, all items with the "latest-vX.Y" tag will be moved to the newer version, even if you didn't make changes them.

5.4 Buttons

The Library Manager contains the following buttons to perform the described actions:

- **Update** – Use this button to update your project with changes made in the Library Manager. This action runs the `make getlibs` command, and you can use it at any time even if you made no changes. You might do this for a project that has no libraries yet or to update the libraries specified as "Latest" to get any updates.
- **Close** – Use this button to close the Library Manager.

- **Retry** – This button displays if the message console indicates that the tool cannot access the manifest file. Use the **Retry** button after adjusting your internet and/or proxy settings to check if the tool can access the manifest file.

5.5 Message console

The area below the BSPs and Libraries displays various messages, such as when you select/deselect an item, click the **Update** or **Retry** button, or select Offline mode.

5.6 Status indicator

At the bottom of the tool, the status indicator displays as faded in offline mode and displays as normal in online mode. If you hover over the indicator, a tooltip displays.

6 Manual library management

If a library is not in a manifest file, it will not appear in the Library Manager and will need to be managed manually. Regardless of the library management flow, this is a two-step process:

- Acquire the library
- Acquire the library's dependencies

These steps can be done several different ways depending on the construction of the library and the library management flow being used by the application. Note that for (custom) BSPs, the Project Creator tool simplifies this process greatly. You can still use the other methods on BSPs if you want to include a custom BSP into an existing application.

Note: *By manually managing libraries, all dependencies will be included directly in the application. Indirect dependencies only occur in the MTB flow when libraries and dependencies are included in manifest files as described in the indirect dependency section.*

6.1 Acquiring a library (MTB flow)

There are three basic ways to get a library that isn't in a manifest file using the MTB flow:

Requirements	Method
Library is hosted in a Git repo	Create a *.mtb* file for the library in the application's *deps* subdirectory.
	Run `make getlibs` from the application's root directory.
	The library can be local or shared based on the *.mtb* file contents.
Any library Library is local to app	Use copy, `git clone`, or use some other revision control system to pull the library into the application. It can be located anywhere except the application's *libs* subdirectory.
Any library Library is in a shared location	Use copy, `git clone`, or use some other revision control system to pull the library into a shared location. Edit the `SEARCH` variable in the application's Makefile to point to the library.
Custom BSP	Use copy, `git clone`, or use some other revision control system to get the BSP somewhere on disk if it isn't already. The location is not important. It will be copied into the application folder during project creation.
	Use the Project Creator **Import** function to select the BSP. The custom BSP will be copied into the application. See the Project Creator user guide for details.

Once you have a library, the next step is to get its dependencies. The methods used depend on whether you are using the MTB flow or the LIB flow.

6.2 Acquiring dependencies (MTB flow)

There are several ways to acquire dependencies:

Requirements	Method
Library contains *.mtbx* files for its dependencies	Run `make import_deps` followed by `make getlibs` from the application's root directory (see import_deps for details).
Dependencies can be local or shared	

Requirements	Method
Library does not contain mtbx files but does contain lib files for its dependencies	Run `make lib2mtbx`, then `make import_deps` and finally `make getlibs` from the application's root directory (see lib2mtbx and import_deps for details).
Dependencies will be local	
Dependencies are in a manifest	Use the Library Manager.
Dependencies are hosted in git repos	Create a *.mtb* file for each dependency in the application's *deps* subdirectory.
	Run `make getlibs` from the application's root directory.
	The dependencies can be local or shared based on the *.mtb* file contents.
Any Dependency	Use copy, `git clone`, or use some other revision control system to pull the library into the application. It can be located anywhere except the application's *libs* subdirectory.
Dependency is local to app	
Any dependency Dependency is in a shared location	Use copy, `git clone`, or use some other revision control system to pull the library into a shared location. Edit the `SEARCH` variable in the application's Makefile to point to the library.
Custom BSP	Dependencies are pulled automatically by Project Creator.

6.2.1 import_deps

The `make import_deps` target is intended specifically to acquire dependencies in MTB flow applications for libraries that don't have their dependencies specified manifest files. To use it, you need a *.mtbx* file in the library for each dependency. The format of *.mtbx* files is identical to *.mtb* files. When you run `make import_deps`, it will copy the *.mtbx* files from the library into the application's deps folder while also changing the extension to *.mtb*. This results in the dependencies being direct to the application which are then pulled in when `make getlibs` is run. The libraries can either be local or shared depending on the contents of the *.mtbx* files. Usage:

```
make import_deps IMPORT_PATH=<path_to_library>
```

6.2.2 lib2mtbx

The `make lib2mtbx` target is mainly intended for libraries (or BSPs) that were created for ModusToolbox™ 2.0 or 2.1 that do not contain *.mtbx* files but do contain lib files for their dependencies. When you run `make lib2mtbx`, it will create *.mtbx* files in the library based on the *.lib* files in the library. The *.mtbx* files will be created to place the dependencies local to the application unless the optional argument “shared” is specified.

You can save the *.mtbx* files to the library's source control so that you don't need to do this step the next time the library is included into an application. Usage:

```
make lib2mtbx CONVERSION_PATH=<path_to_library> [CONVERSION_TYPE="shared"]
```

6.3 Acquiring a library (LIB flow)

There are two basic ways to get a library that isn't in a manifest file using the LIB flow.

Requirements	Method
Library is hosted in a git repo	Create a *.lib* file for the library in the application's *deps* subdirectory.
	Run `make getlibs` from the application's root directory.
Any library	Use copy, `git clone`, or use some other revision control system to pull the library into the application. It can be located anywhere except the application's *libs* subdirectory.

Requirements	Method
Custom BSP	Use copy, `git clone`, or use some other revision control system to get the BSP somewhere on disk if it isn't already. The location is not important. It will be copied into the application folder during project creation.
	Use the Project Creator **Import** function to select the BSP. The custom BSP will be copied into the application. See the Project Creator user guide for details.

6.4 Acquiring dependencies (LIB flow)

Again, there are several ways to acquire dependencies depending on the library's structure:

Requirements	Method
Library contains *.lib* files for its dependencies	Run `make getlibs` (this step is already done if you used `make getlibs` to acquire the library).
Dependencies are listed in a manifest	Use the Library Manager.
Dependencies are hosted in a git repo	Create a *.lib* file for each dependency in the application's *deps* subdirectory.
	Run `make getlibs` from the application's root directory.
Any dependency	Use copy, `git clone`, or use some other revision control system to pull the library into the application. It can be located anywhere except the *libs* directory.
Custom BSP	Dependencies are pulled automatically by Project Creator.

7 Tool change description

This section lists and describes the changes for each version of this tool.

Version	Change descriptions
1.0	New tool.
1.1	Added Settings and Help menus. Moved link about version changes to the message console, when it is applicable. Added icon to indicate online/offline status. Removed Summary dialog; summary is shown in the console.
1.2.0	Added Retry button. Changed Apply button to Update. Added Close button. Tool can be launched from the Windows Start menu. Updated versioning to support patch releases. Updated for the MTB flow. Added toolbar commands to show/hide items, as well as collapse and expand the trees.
1.30	Added indications for newer versions of BSPs and libraries. For BSPs and libraries that are not selected, the displayed version was changed from Latest #.X release to the actual most recent X.Y.Z release.
1.40	Updated the handling on .mtbx files.

Revision history

Revision	Date	Description
**	10/16/19	New document.
*A	10/17/19	Added a warning about removing all Cypress BSPs/libraries when there is no custom BSP.
*B	03/26/20	Updated to version 1.1.
*C	09/01/20	Updated to version 1.2.0.
*D	10/07/20	Added details for BTSDK 2.8.
*E	03/25/21	Updated to version 1.30.
*F	09/28/21	Updated to version 1.40.

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

Edition 2021-09-28
Published by
Infineon Technologies AG
81726 Munich, Germany

Do you have a question about this document?
Email: erratum@infineon.com

Document reference
002-28736 Rev. *F

IMPORTANT NOTICE
The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics (“Beschaffenheitsgarantie”) .

With respect to any examples, hints or any typical values stated herein and/or any information regarding the application of the product, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation warranties of non-infringement of intellectual property rights of any third party.

In addition, any information given in this document is subject to customer’s compliance with its obligations stated in this document and any applicable legal requirements, norms and standards concerning customer’s products and any use of the product of Infineon Technologies in customer’s applications.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer’s technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on the product, technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies office (**www.infineon.com**).

WARNINGS
Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies’ products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.