# **Commands**

## mtbassist.mtbCreateProject

This command starts the ModusToolbox project creator GUI to create a new project and then loads the project into visual studio code.

Arguments: None

Command Palette: Yes

* If a current application is loading, display an error message and quit command
* Start ModusToolbox project creator GUI
* When project creator GUI is complete
  + If no project is created, tell the user no project was created
  + If a single application is created, load the application (see section 7)
  + If multiple applications are created, prompt for the one to load, then load the selected application (see section 7).

## mtbassist.mtbImportProject

This command displays a web view describing what the import command does. It then provides a link at the bottom of the web view page to run the mtbassist.mtbImportProjectDirect command which does the actual import.

Arguments: None

Command Palette: Yes

* Display the import web view page describing the operation

## mtbassist.mtbImportProjectDirect

This command clones an existing project from a git repository and then readies it for use on the local machine. Specifically, it does a “git clone” followed by a “make getlibs” followed by a “make vscode” command.

Arguments: None

Command Palette: Yes

* If a current application is loading, display an error message and quit command
* Prompt the user for a location on the local disk for the project directory (LOCALDIR)
* Prompt the user for the URL of a git repository to clone (REPOPATH)
* Prompt the user for the name of the project directory locally (LOCALNAME)
* If the LOCALDIR does not exist, display an error and quit command
* If the LOCALDIR exists, but is not a directory, display an error and quit command
* Use git from modus-shell package to clone REPOPATH to LOCALDIR/LOCALNAME, logging output into ModusToolbox log window. If git clone fails display message and quit command
* Use make, from modus-shell package, to run ‘make getlibs’ in the directory LOCALDIR/LOCALNAME, logging output into ModusToolbox log window. If make fails display message and quit command.
* Use make, from modus-shell package, to run ‘make vscode’ in the directory LOCALDIR/LOCALNAME, logging output into ModusToolbox log window. If make fails display message and quit command.
* Open the application directory LOCALDIR/LOCALNAME In vscode (see section 7).

## mtbassist.mtbRunEditor

This command is an internal command used to run an editor (library manager, configurator, etc.) in the current application directory.

Arguments: MTBLaunchConfig data structure

Command Palette: No

* Displays a message stating the program is running with the short name of the program from the output of mtblaunch.
* If the program cannot be launched, display an error message and put error message in the ModusToolbox log window, then quit the command.
* Any standard output or standard error output by the program should be displayed in the ModusToolbox log window.
* When the command quits, if an ModusToolbox application is loaded, it should be reloaded (see section 7)

## mtbassist.mtbShowDoc

This command is an internal command used to display a document as defined by the mtblaunch program.

Arguments: MTBLaunchDoc data structure

Command Palette: No

* Displays a message stating that the document requested is being loaded.
* Displays the document requested, opened per the conventions of the local operating system.

## mtbassist.mtbShowWelcomePage

This command shows the ModusToolbox Assistant welcome page. See section 6 for more details.

Arguments: None

Command Palette: Yes

## mtbassist.mtbTurnOnDebugMode

This command turns on debug mode for the ModusToolbox Assistant plugin. All debug messages will be displayed to the ModusToolbox log window. This setting is a persistent setting and should remain across invocations of visual studio code and across different workspaces.

Arguments: None

Command Palette: Yes

## mtbassist.mtbTurnOffDebugMode

This command turns off debug mode for the ModusToolbox Assistant plugin. No debug messages will be displayed to the ModusToolbox log window. This setting is a persistent setting and should remain across invocations of visual studio code and across different workspaces.

Arguments: None

Command Palette: Yes

## mtbassist.mtbRunLibraryManager

This command runs the library manager in the currently loaded ModusToolbox application.

Arguments: None

Command Palette: Yes

* If there is no ModusToolbox application loaded, display an error message, and quit command.
* If a ModusToolbox application is actively loading, display an error message, and quit command.
* If a ModusToolbox application is loaded, but is not valid, display an error message, and quit command.
* If the library-manager was not reported from ModusToolbox in mtblaunch, display an error message, and quit command.
* Run the library manager using the mtbRunEditor command

## mtbassist.mtbSymbolDoc

This command looks at the symbol under the cursor in the active text editor and if the file that contains the definition of the symbol is in an asset provided by ModusToolbox, then the documentation for that asset is displayed.

Arguments: None

Command Palette: Yes

* Execute the vscode.executeDefinitionProvider for the current position in the current document to get the definition of the symbol under the cursor.
* If the result of the vscode.executeDefinitionProvider command is empty, display a message and quit the command.
* If the result is not empty, find an asset associated with filename that contains the symbol definition.
* If an asset was not found, display a message, and quit the command.
* If an asset was found, but the asset does not contain documentation, display a message and quit the command.
* If an asset was found and contains documentation, open all documents associated with the asset found (may be more than one).

# **Key Bindings**

## ctrl+shift+F1

Runs the mtbassist.mtbSymbolDoc command in the current window.

# **Views**

## mtbprojinfo

This view will be named PROJECT INFO and will contain the application name, the target, the device, any additional devices, and any defined components. For applications with multiple projects, there will be a section per project.

## mtbglobal

This view will be named TOOLS and will contain three sections which are global, application, and project. The tools displayed will be those reported by mtblaunch for the current project and they will be placed into sections based on the information provided by mtblaunch.

## mtbdocs

This view will be named DOCUMENTATION and will contains the top level landing page for ModusToolbox form the installation directory and a section called Application which contains all of the documents reported by mtblaunch for the assets in the current application.

## mtbassets

This view will be named ASSETS and will list all of the assets used by the current

# **Menus**

## mtbSymbolDoc

* If a “C” file is loaded in the editor, the command ‘ModusToolbox Documentation’ should be present in the right click context menu.
* If the menu item is selected, there are one of two outcomes.
  + If the symbol under the cursor is part of a ModusToolbox asset, documentation for that asset is displayed.
  + If the symbol under the cursor is not part of an asset, a message is displayed indicating that the symbol is not part of an asset.

# **Terminal**

## ModusToolbox

The ModusToolbox Shell is integrated into visual studio code as a terminal. It is created by selecting the terminal tab in the bottom window and then using the arrow

# **ModusToolbox Assistant Welcome Page**

asdf

# **Open ModusToolbox Application Folder**