EZ-PD™ PMG1-S3 Dock SDK user guide

About this document

Scope and purpose

This guide provides information and instructions for using the EZ-PD™ PMG1-S3 Dock Software Development Kit (SDK). The PMG1-S3 Dock SDK is a software solution that allows users to harness the capabilities of the Type-C and PD controllers from Infineon to create an integrated Dock management controller and Power Delivery controller solutions for docking stations.

Intended audience

This document helps the application developers to understand all the APIs and the tools included with the PMG1-S3 Dock SDK.

Document conventions

Table 1 Document conventions

Convention	Explanation
Bold	Emphasizes heading levels, column headings, table and figure captions, screen names, windows, dialog boxes, menus, and sub-menus
Italics	Denotes variable(s) and reference(s)
`Courier New`	Denotes APIs, functions, interrupt handlers, events, data types, error handlers, file/folder names, directories, command-line inputs, code snippets
>	Indicates that a cascading sub-menu opens when you select a menu item

Table of contents

Table of contents

1 Introduction

1.1 USB Type-C and power delivery

USB Type-C is the new USB-IF standard that solves several challenges faced by today's Type-A and Type-B cables and connectors. USB Type-C uses a slimmer connector (measuring only 2.4 mm in height) to enable increasing miniaturization of consumer and industrial products. The USB Type-C standard is gaining rapid support by enabling small form-factor, easy-to-use connectors, and cables that can transmit multiple protocols. In addition, it offers power delivery up to 240 W – a significant improvement over the 7.5 W for previous standards.

1.1.1 USB Type-C highlights

- The new reversible connector measuring only 2.4 mm in height.
- Compliant with USB Power Delivery Specification, providing up to 240 W.
- Increases the data bandwidth to 40 Gbps with USB4.
- Combines multiple protocols in a single cable, including DisplayPort, PCIe, and Thunderbolt.

1.2 USB-C high-voltage microcontrollers

EZ-PD™ PMG1 (Power Delivery microcontroller Gen 1) is the industry's first family of high-voltage microcontrollers (MCU) with USB-C Power Delivery (PD). These chips include an Arm® Cortex®-M0/M0+ CPU and a USB-C PD controller along with analog and digital peripherals. It is targeted to any embedded system that provides/consumes power to/from a high-voltage USB-C PD port and leverages the microcontroller to provide additional control capability.

In addition to the EZ-PD™ PMG1-S3 which is used as a PD controller and a Dock Management Controller (DMC) in dock solutions, the following devices from Infineon can also be used as PD controllers in dock solutions:

- EZ-PD™ CCG7SC
- EZ-PD™ CCG7DC

For more details on other Type-C controllers of Infineon and for feature comparison, see the USB-C Power Delivery controllers.

1.3 PMG1-S3 Dock SDK

The PMG1-S3 Dock SDK is a collection of software libraries and reference projects hosted on GitHub repos and made available through the ModusToolbox™ software development environment.

The PMG1-S3 Dock SDK supports the following reference projects:

- PMG1-S3-based Griffin Creek Dock solution supports DP, USB4, and TBT alternate modes.
- PMG1-S3-based USB-C Dock solution supports DP alternate mode.

The SDK provides a firmware stack compatible with Type-C and USBPD specifications, along with the necessary drivers, and software interfaces required to implement Dock applications using the PMG1-S3 Power Delivery controllers.

The key features of PMG1-S3 Dock controller solutions are:

- Type-C port configuration and connection detection in sink, source, and dual role on Port 0 and source role on Port 1
- USB PD communication with USB PD revision 3.1 v1.8 compliance
- Supports extended power range (EPR) in source role on Port 0
- Supports VESA Display Port Alt Mode on Type-C standard v1.0 that is compatible with DP specifications 1.3 and 1.4
- USBFS device support (vendor, HID, and Billboard classes). HID class is available only for PMG1-S3 USB-C Dock solution.
- Supports power protections, such as:
  - VBUS overvoltage protection (VBUS OVP)
  - VBUS overcurrent protection (VBUS OCP)
  - VBUS short-circuit protection (VBUS SCP)
  - VCONN overcurrent protection (VCONN OCP)
  - VBUS undervoltage protection (VBUS UVP)
- Supports Dock management controller features, such as:
  - Signed/unsigned offline firmware update for PMG1-S3 and onboard components such as CCG7SC for PMG1-S3 USB-C Dock solution and Goshen Ridge, Foxville for PMG1-S3 Griffin Creek Dock solution.
  - Smart power management
  - Supports storing and retrieving ETAG information. This feature is available only for PMG1-S3 USB-C Dock solution.
- Buck-boost controller drivers for MPS4247 and RichTek6190 controllers.
- Host Processor Interface (HPI) master implementation for communication with EZ-PD™ CCG7SC slave device. This feature is available only for PMG1-S3 USB-C Dock solution.
- UART and internal flash logging mechanism for debugging
- LED control (for example, ON, OFF, blink, and breath) driver for custom use cases in the dock design.
- Adapter detection logic
- Deep Sleep operation
- Supports I2C slave for notifying the Intel TBT controller to query PMG1-S3 when PD state change occurs

1.3.1 SDK components

1.3.1.1 Reference projects

The SDK includes reference projects for the target applications that is used to obtain a jump-start in the process of developing PMG1-S3-based Dock Management Controller applications.

- mtb-example-pmg1s3-griffin-creek-dock

  This project implements a dual Type-C port DMC for Thunderbolt applications using the PMG1-S3 device. The project also supports USB4 and DP alternate modes.

- mtb-example-pmg1s3-usbc-dock

  This project implements a dual Type-C port DMC for display port applications using the PMG1-S3 device.

- mtb-example-ccgx-usbc-dock

  This firmware running on CCG7SC controls one of the source-only downstream ports connected to the dock to configure the external data path muxes for routing the USB 5Gbps lanes appropriately in the Dock hardware. CCG7SC supports the HPI Slave feature that allows the PMG1-S3 (an HPI master) to monitor, control the port behavior and update CCG7SC firmware.

  This reference solution is released in the binary form in the Infineon Developer Center.

1.3.1.2 Middleware libraries

The SDK uses the following middleware libraries to implement the key features supported by the dock solution:

1. PDStack middleware library

   The PDStack middleware implements state machines defined in the USB Type-C cable and connector specification and the USB PD specification. The middleware provides a set of device policy manager (DPM) APIs through which the application can initialize, monitor, and configure the middleware operations.

2. PDAltmode middleware library

   The PDAltMode middleware implements state machines to work with different USB PD alternative modes. The middleware provides a set of PDAltMode APIs through which the application can initialize, monitor, and configure the PD alternative mode operation.

3. PDUtils middleware library

   The PDUtils middleware implements the software timer and utility functions required by EZ-PD™ PMG1/CCGx/WLC devices.

4. USBdevice middleware library

   The USB device middleware provides a Full speed USB 2.0 Chapter 9 specification compliant device framework. It uses the USBFS driver from the CAT2 peripheral driver library to interface with the hardware. The middleware provides support for audio, CDC, HID, and vendor classes. It also enables implementing support for other classes. The USB configurator tool makes it easy to construct a USB Device descriptor.

5. Host processor interface (HPI) middleware library

   The HPI middleware implements an HPI master interface to monitor or control the operating condition of other HPI slave devices. The middleware implements asynchronous interrupt-based event queue handling.

6. mtb-pdl-cat2 peripheral driver library

   The mtb-pdl-cat2 peripheral device library (PDL) simplifies the software development for the PSoC™ 4 and PMG1 (CAT2) family devices. The PDL integrates device header files, startup code, and peripheral drivers into a single package.

7. PmgAppCommon middleware library

   The PmgAppCommon middleware provides the following functionalities that are essential for a USB-C and Power Delivery application:

   - Application-level handler for PDStack events
   - Provides the application status information
   - Prepares application layer for device low-power mode
   - Evaluates the source capabilities advertised by a port partner and identify the optimal power contract to be entered
   - Evaluates a PD request data object and determines whether to accept or reject the request
   - Evaluates a power role (PR) swap, data role (DR) swap, or a VConn role swap request from a port partner
   - Enables/disables the power source output
   - Sets the desired voltage and current for the power source output
   - Enables/disables the power sink path
   - Handler for voltage/current change
   - Initializes the vendor defined messages (VDM) handler
   - Evaluates the received VDM messages and respond to them
   - Enables/disables fault protections such as VBUS OVP, VBUS OCP, VBUS SCP, VBUS UVP, and VConn OCP.
   - Default handlers for fault protections
   - Handles enumeration of USB Billboard interface and USB vendor commands
   - Controls external buck-boost regulators
   - Supports firmware update for various devices within the dock
   - Smart power management
   - UART and flash logging for debugging

2 SDK dependencies

2.1 Tool dependencies

2.1.1 ModusToolbox™

Infineon's Type-C controllers are based on Infineon's PSoC™ 4 programmable system-on-chip architecture, which includes programmable analog and digital blocks, an Arm® Cortex®-M0 core, and internal flash memory.

The Eclipse IDE that comes as part of ModusToolbox™ is used to configure PMG1 devices, to develop and compile the firmware applications and optionally to program the devices using SWD. This SDK version requires ModusToolbox™ v3.1 or higher.

This ModusToolbox™ version can be installed and used on a computer along with previous versions of ModusToolbox™.

ModusToolbox™ include configuration tools, low-level drivers, libraries, and operating system support, most of which are compatible with Linux, macOS, and Windows-hosted environments. For more details, see Getting Started with ModusToolbox™.

The following sections briefly describe the ModusToolbox™ tools needed for the dock solutions.

2.1.1.1 Project creator

The ModusToolbox™ software includes the Project Creator as both a GUI tool and a command-line tool to create ModusToolbox™ applications. The Project Creator tool clones the selected BSP and code example template(s), and then creates the directory structure at the specified location with the specified name. The Project Creator tool also runs the required processes to download and import all the necessary libraries and dependencies. For more details, see the ModusToolbox™ Project Creator user guide.

2.1.1.2 Device configurator

The Device configurator is used to set up the system (platform) functions such as pins, interrupts, clocks, and DMA, as well as the basic peripherals, including UART, Timer, and so on. For more details, see the ModusToolbox™ Device Configurator user guide.

2.1.1.3 USB configurator

Configure the USB settings to generate the required firmware. This includes options for defining the device descriptor and settings. For more details, see the ModusToolbox™ USB Configurator user guide.

2.1.1.4 Python

The reference projects make use of a set of Python scripts to customize the binaries at the end of the build process. These Python scripts use the Python that is installed as part of the ModusToolbox™ software environment.

2.1.2 ModusToolbox™ programmer

The ModusToolbox™ programmer is a standalone, cross-platform, flash programmer tool that provides a graphical user interface to program, erase, verify, and read the flash of the target device. It is delivered with the ModusToolbox™ programming tools package, and it supports HEX, SREC, ELF, and BIN programming file formats.

ModusToolbox™ Programmer version 5.0 is used to program, erase, verify, and read the flash of PMG1-S3 in this version of Dock SDK. For more details, see the ModusToolbox™ Programmer GUI user guide.

2.1.3 EZ PD™ Dock tools

2.1.3.1 EZ-PD™ Dock Configuration Utility

The EZ-PD™ Dock Configuration Utility is a GUI application that supports PMG1-S3-based Dock designs.

The utility supports Dock configuration for PMG1-S3 USB-C and PMG1-S3 Griffin Creek Dock solutions, image creation, and signature update to create a composite binary file for updating the Dock components.

2.1.3.2 EZ-PD™ firmware update utility

The EZ-PD™ firmware update utility is a command-line application (tool) that runs on Windows systems. The tool uses the WinUSB driver to communicate with the dock. The tool updates the device firmware in the dock and reports the consolidated status. This tool takes a composite image (consists of FWCT table, signature data (applicable only for signed firmware update mode) and firmware images of PMG1-S3 and connected components) as input and programs it on the external SPI Flash through the PMG1-S3. This tool supports both signed and unsigned firmware updates.

Both the utilities are packaged in a single installation package (i.e., *EZ-PD_Dock_Configuration_Utility_Setup.exe*), which can be downloaded from Infineon Developer Center.

For more details on the utilities, see the EZ-PD™ Dock Configuration Utility user guide that is installed along with the utilities using the installation package.

2.2 Hardware dependencies

The PMG1-S3 USB-C Dock board can be used to evaluate the PMG1-S3 USB-C Dock solution.

The Griffin Creek dock reference design can be used to evaluate the PMG1-S3 Griffin Creek Dock solution

For more details on the hardware files and EZ-PD™ PMG1-S3 Dock user guide, see the EZ-PD™ PMG1-S3 Dock SDK.

3 Getting started with PMG1-S3 Dock SDK

3.1 Preparing to use ModusToolbox™

3.1.1 Environment updates

The ModusToolbox™ tools are located at *<install_path>/ModusToolbox/tools_MAJOR.MINOR*.

If multiple versions of ModusToolbox™ are installed on the PC, multiple tools will be present in the *<install_path>/ModusToolbox/* location. Therefore, use the **CY_TOOLS_PATHS** system variable to have the required set of tools. On Windows, open the **Environment Variables** dialog, and create a new System/User Variable (see Figure 1).

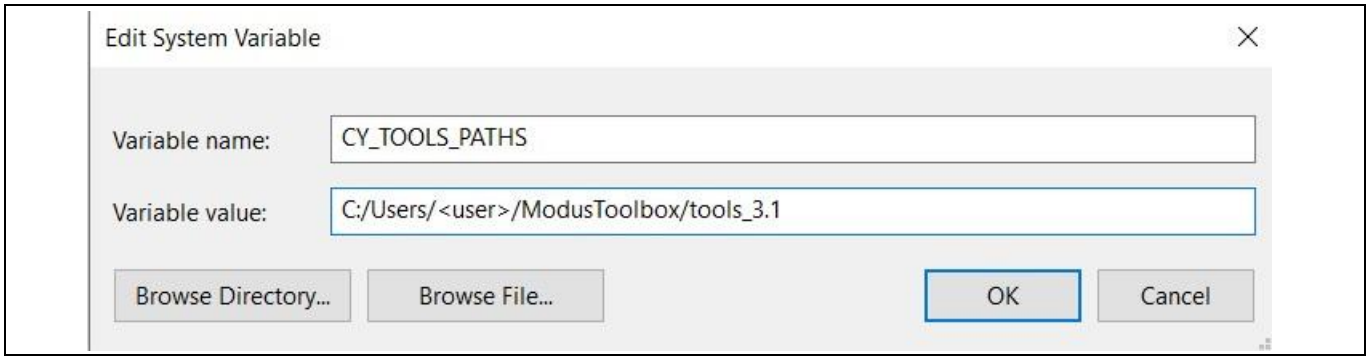

Figure 1 Environment variable updates

*Note: For **Variable value**, use a Windows style path (that is, not like /cygdrive/c/). For example, C:/Users/<user>/ModusToolbox/tools_3.1/*

Use the correct method for setting variables in macOS and Linux for your system.

3.2 Using the reference projects

Each reference project provided with this SDK can be imported into Eclipse IDE for ModusToolbox™, customized as per the user requirements and compiled. This section describes steps for project creation, compiling, programming, and debugging for EZ-PD™ PMG1-S3 USB-C Dock solution. The same steps can be used for creating, compiling, programming, and debugging the EZ-PD™ PMG1-S3 Griffin Creek Dock solution as well.

Note: These projects are designed to work with the specific devices that are mentioned earlier. Changing the target device using the library manager can fail the firmware build.

3.2.1 Project creation

3.2.1.1 In Eclipse IDE for ModusToolbox™ software

1. Launch the Eclipse IDE for ModusToolbox™ software and select the workspace, as shown in Figure 2.

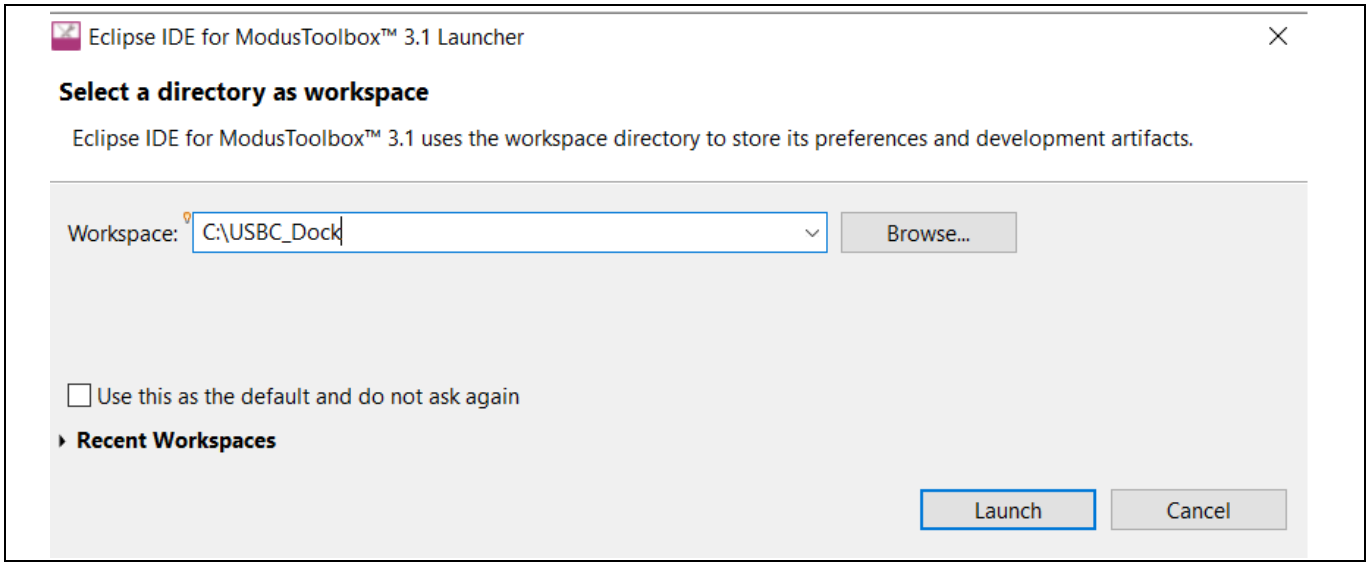

Figure 2 Launching Eclipse IDE for ModusToolbox™ software

2. Click **New Application** in the **Quick Panel** (or use **File > New > ModusToolbox™ Application**), as shown in Figure 3. This launches the Project Creator tool.

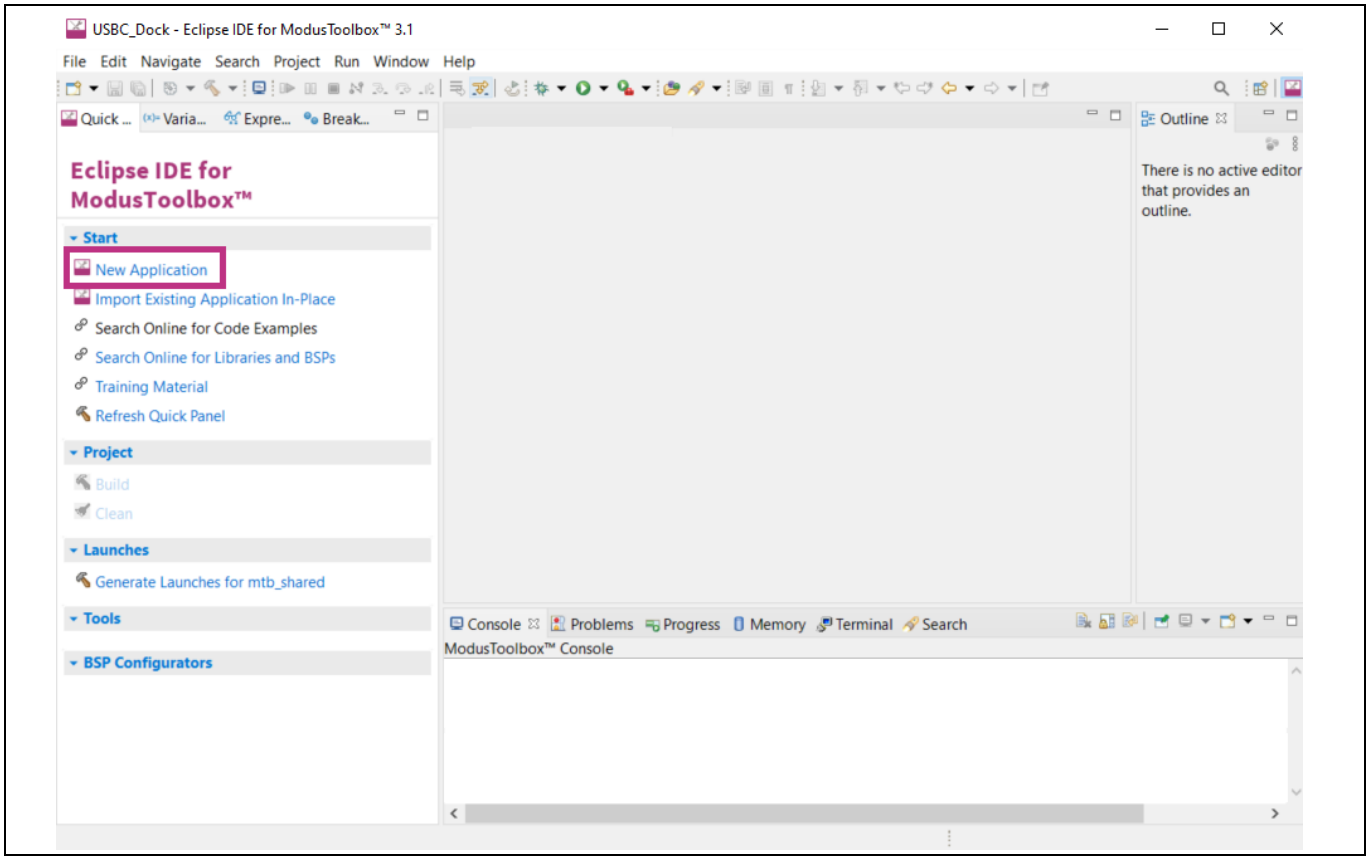

Figure 3 Creating New Application

3. In **Choose Board Support Package (BSP) – Project Creator 2.10** window, do the following:
   a) Expand the **PMG BSPs** drop-down.
   b) Select **PMG1S3DUAL.**
   c) Click **Next**.

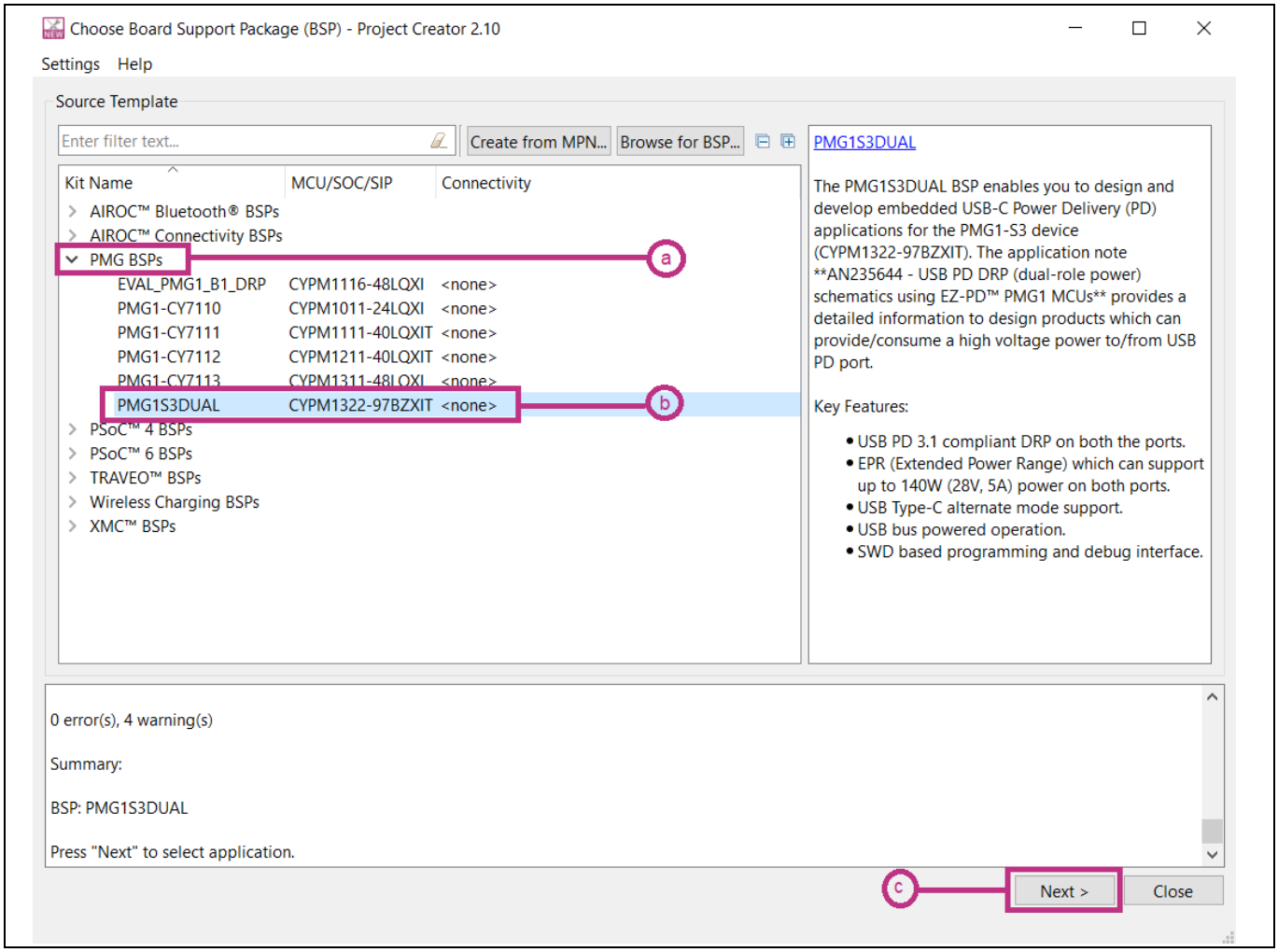

Figure 4 Selection of BSP

4. Do the following in the next window (see Figure 5):
   a) Expand the **Peripherals** drop-down.
   b) Select the application.
   c) Click **Create**.

   For example, in Figure 5, PMG1-S3 USB-C Dock Solution is selected and created.

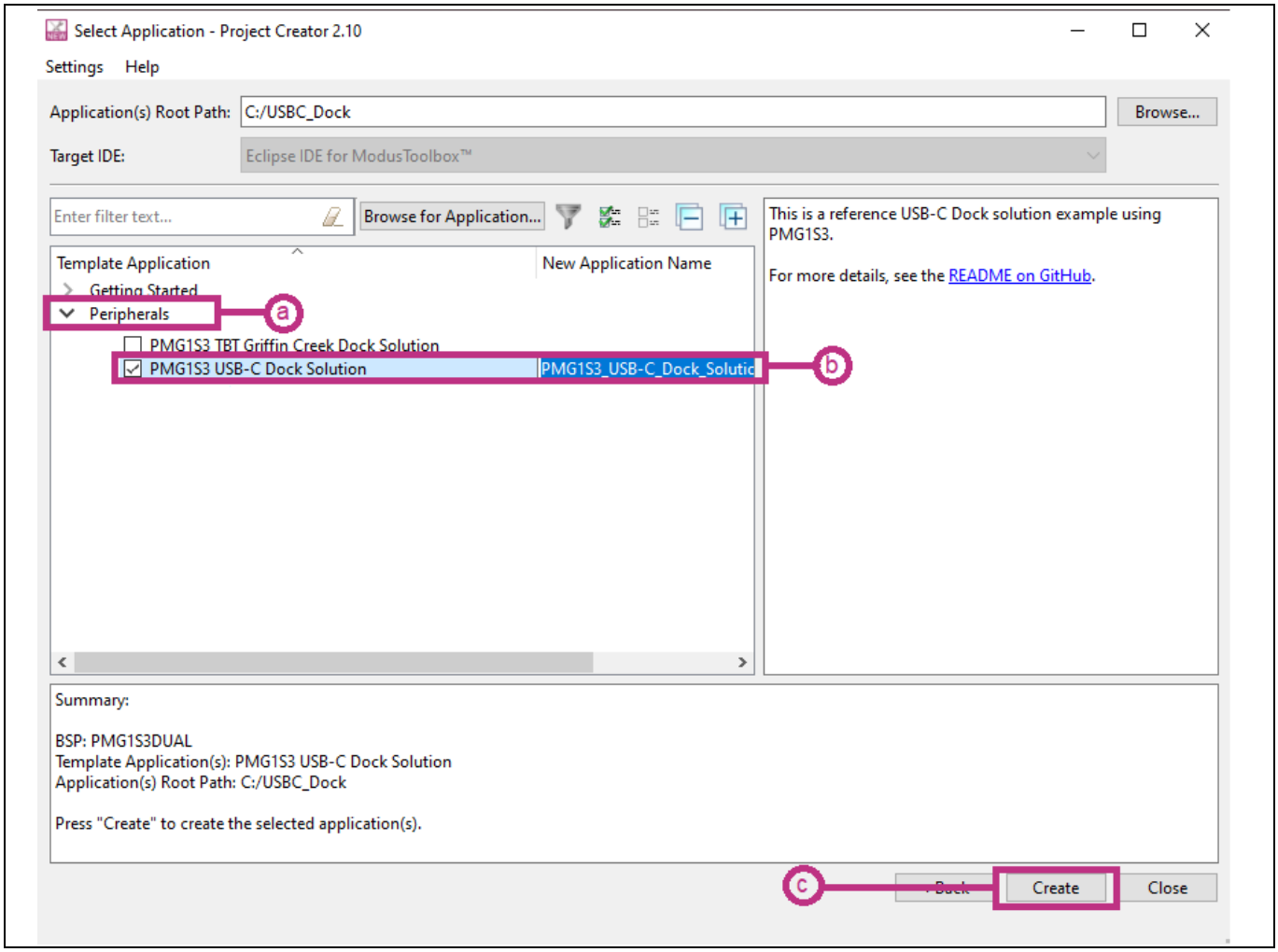

Figure 5 Selection of reference project

*Note: To create PMG1-S3 Griffin Creek Dock Solution, select "PMG1S3 TBT Griffin Creek Dock Solution" shown in the above figure and click **Create**.*

5. The project creator GUI will close automatically if the project creation is successful or if it encountered an error in between creation of the project. If the project creation was successful, then the project will be available in the Eclipse IDE for ModusToolbox™ to modify, compile, build, and debug.

3.2.1.2 In command-line interface (CLI)

The ModusToolbox™ software provides the Project Creator as both a GUI tool and the command-line tool, “project-creator-cli”. The CLI tool is used to create applications from a CLI terminal or from within batch files or shell scripts. This tool is available in the *{ModusToolbox™ software install directory}/tools_{version}/project-creator/* directory.

Use a CLI terminal to invoke the “project-creator-cli” tool. On Windows, use the command-line “modus-shell” program provided in the ModusToolbox™ software installation instead of a standard Windows command-line application. This shell provides access to all ModusToolbox™ software tools. To access it type `modus-shell` in the search box in the Windows menu. In Linux and macOS, use any terminal application.

Table 2 lists the arguments of the “project-creator-cli” tool.

Table 2 Arguments for project-creator-cli

Argument	Description	Required/optional
`--board-id`	Defined in the `<id>` field of the BSP manifest.	Required
`--App-id`	Defined in the `<id>` field of the CE manifest.	Required
`--target-dir`	Specify the directory in which the application is to be created if you prefer not to use the default current working directory.	Optional
`--user-app-name`	Specify the name of the application if you prefer to have a name other than the example’s default name.	Optional

On Windows, run the following command in the modus-shell to clone the PMG1-S3 USB-C Dock solution with the desired name “USBCDock”, to the *C:/mtb_dock_projects* directory:

```
project-creator-cli --board-id PMG1S3DUAL --app-id mtb-example-pmg1s3-usbc-dock --
user-app-name USBCDock --target-dir "C:/mtb_dock_projects"
```

After the successful execution of the command, the “USBCDock” project will be created inside the directory *C:/mtb_dock_projects*.

On Windows, run the following command in the modus-shell to create the EZ-PD™ PMG1-S3 Griffin Creek Dock Solution with the desired name “TBTDock” to the *C:/mtb_dock_projects* directory:

```
project-creator-cli --board-id PMG1S3DUAL --app-id mtb-example-pmg1s3-griffin-
creek-dock --user-app-name TBTDock --target-dir "C:/mtb_dock_projects"
```

3.2.2 Compiling the reference project

This section details the steps to be followed to compile the dock reference projects.

PMG1-S3 USB-C Dock solution and PMG1-S3 Griffin Creek Dock solution architectures use the asymmetric dual firmware images with a bootstrap. For the flash memory map and for more information on firmware operation, see the Firmware architecture.

The Makefile present in the application is designed to disable/enable the certain features during the build process for firmware 1 (fw1) and firmware 2 (fw2) respectively.

For more information on the compile-time options, see the Compile-time options and Module operation structure.

To start the build operation for firmware 1, use either of the steps described in the Using Eclipse IDE for ModusToolbox™ software and Using CLI sections. A post-build script (*post_build.sh*) is executed at the end of firmware 1 build. The steps outlined in the Post-build processing section will be performed by the post-build script.

3.2.2.1 Using Eclipse IDE for ModusToolbox™ software

The steps to compile the project using the Eclipse IDE for ModusToolbox™ software are:

1. In **Project Explorer**, select the application (PMG1S3_USB-C_Dock_Solution in this case)
2. In **Quick Panel**, select **Build Application,** as shown in Figure 6.

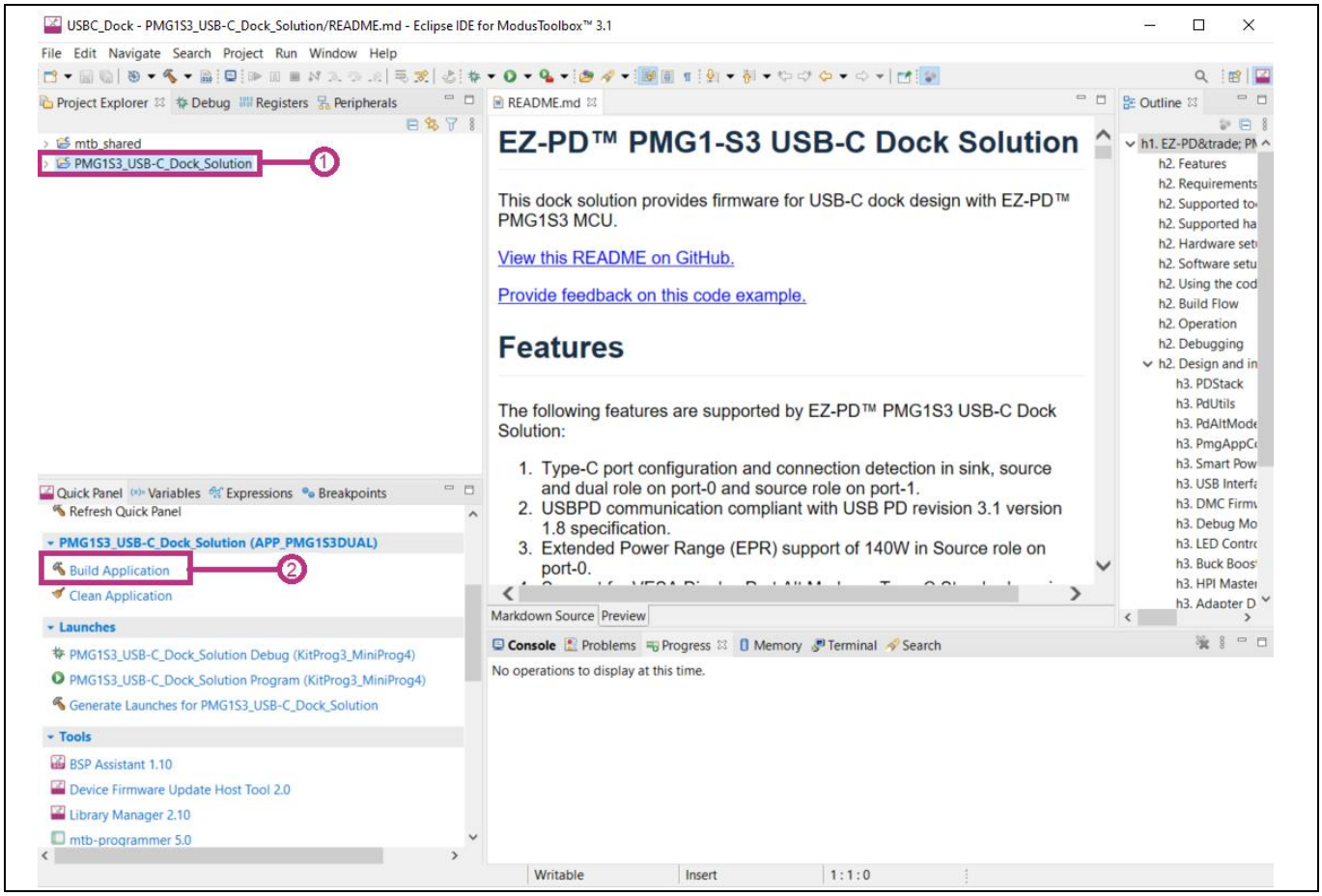

Figure 6 Building application using Eclipse IDE for ModusToolbox™ software

3.2.2.2 Using CLI

The steps to compile the project using the CLI are:

1. Open terminal and navigate to the application directory.
2. Run the following command:

```
make build -j8
```

3.2.2.3 Post-build processing

The post-build script executes the following steps:

1. On completion of the fw1 build, the post-build script is executed. This generates a *.cyacd2* file for Firmware 1 (fw1) and also triggers the Firmware 2 (fw2) build operation.
2. After the Firmware 2 build operation is complete, the post-build script is reinvoked to generate the *.cyacd2* file for Firmware 2 and perform the following steps to create a combined HEX file.
   a) Extract the silicon ID from the fw2 hex file generated by the build process.
   b) Create an intermediate combined HEX file by combining the Bootstrap, firmware-1, and firmware-2 using the cymcuelftool. The cymcuelftool is installed along with ModusToolbox™. User guide for cymcuelftool is found in the following location:

      C:/Users/<user>/ModusToolbox/tools_3.1/cymcuelftool-1.0/docs

   c) Convert the intermediate combined HEX file to a temporary combined binary file (*.bin*) using the srec_cat tool. The srec_cat tool is installed along with ModusToolbox™. Documents related to srec_cat tool is found in the following location:

      C:/Users/<user>/ModusToolbox/tools_3.1/srecord/docs

   d) Use the bin2psochex tool (see the PMG1-S3 USB-C Dock and PMG1-S3 Griffin Creek Dock sections for more details) to set the write protection for the Bootstrap flash region and convert the temporary combined binary file back to an intermediate PSoC™-formatted combined HEX file.
   e) Calculate the HASH using SHA-256 for both firmware images (fw1 and fw2) from the intermediate PSoC™-formatted combined HEX file and appending it to the respective metadata flash row to generate the final PSoC™-formatted combined HEX file. The HASH will be used by the Bootstrap (see the Bootstrap) to validate the firmware binary integrity before starting the application.

3.2.2.4 Files generated by post-build script

The post-build script generates the following files:

- ***.cyacd2*:** The post-build script generates two *.cyacd2* files, one for fw1 and another for fw2. These files are used by the firmware update tool to create a composite image for firmware update.

  PSoC™-formatted combined HEX: This HEX file contains the bootstrap, fw1, and fw2 images. This hex file is used for programming PMG1-S3 on the dock using the SWD interface (see the Programming PMG1-S3 on the dock section).

3.2.3 Programming PMG1-S3 on the dock

This section describes the methods that can be used for programming the HEX file generated by building the project to the target that is, PMG1-S3. The HEX file generated will be in the following location within the application directory:

build/APP_PMG1S3DUAL/CONFIG/APPNAME.hex

where,

- "CONFIG" can be debug, release or custom depending on the makefile setting.
- "APPNAME" is the name of the application that was compiled. It will be "mtb-example-pmg1s3-usbc-dock" for the EZ-PD™ PMG1-S3 USB-C Dock solution. For EZ-PD™ PMG1-S3 Griffin Creek Dock solution, it will be "mtb-example-pmg1s3-griffin-creek-dock".

3.2.3.1 Using mtb-programmer

1. Connect the MiniProg4 to the programming header on the board.
2. Open ModusToolbox™ Programmer (mtb-programmer).
3. When the mtb-programmer starts running, do the following (see Figure 7):
   a) Select the device name in the **Probe/Kit** drop-down.
   b) Set the **Platform** to **PMG1** if it is not set.

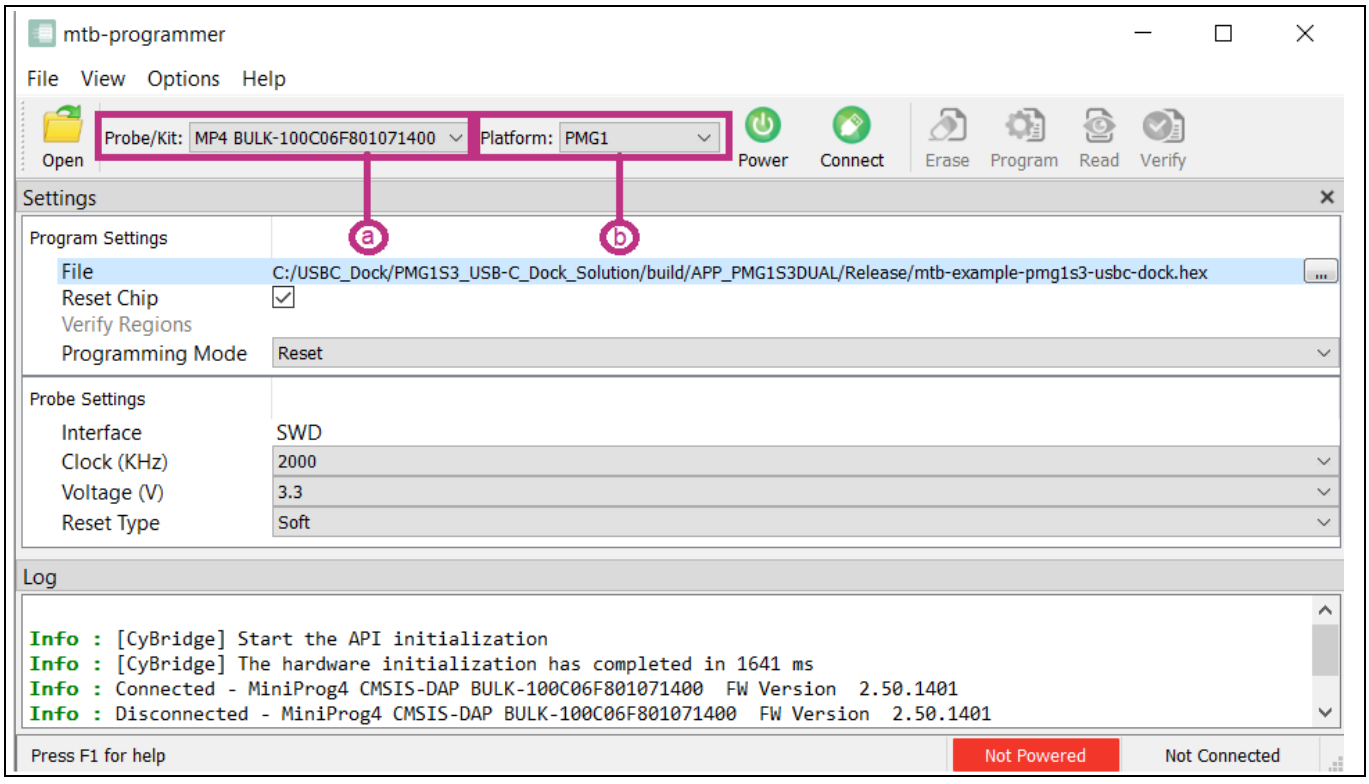

Figure 7 Setting probe/kit and platform settings for mtb-programmer

4. If the device is not powered, the status message “Not Powered” is displayed in the status bar. Click **Power** to power up the device.

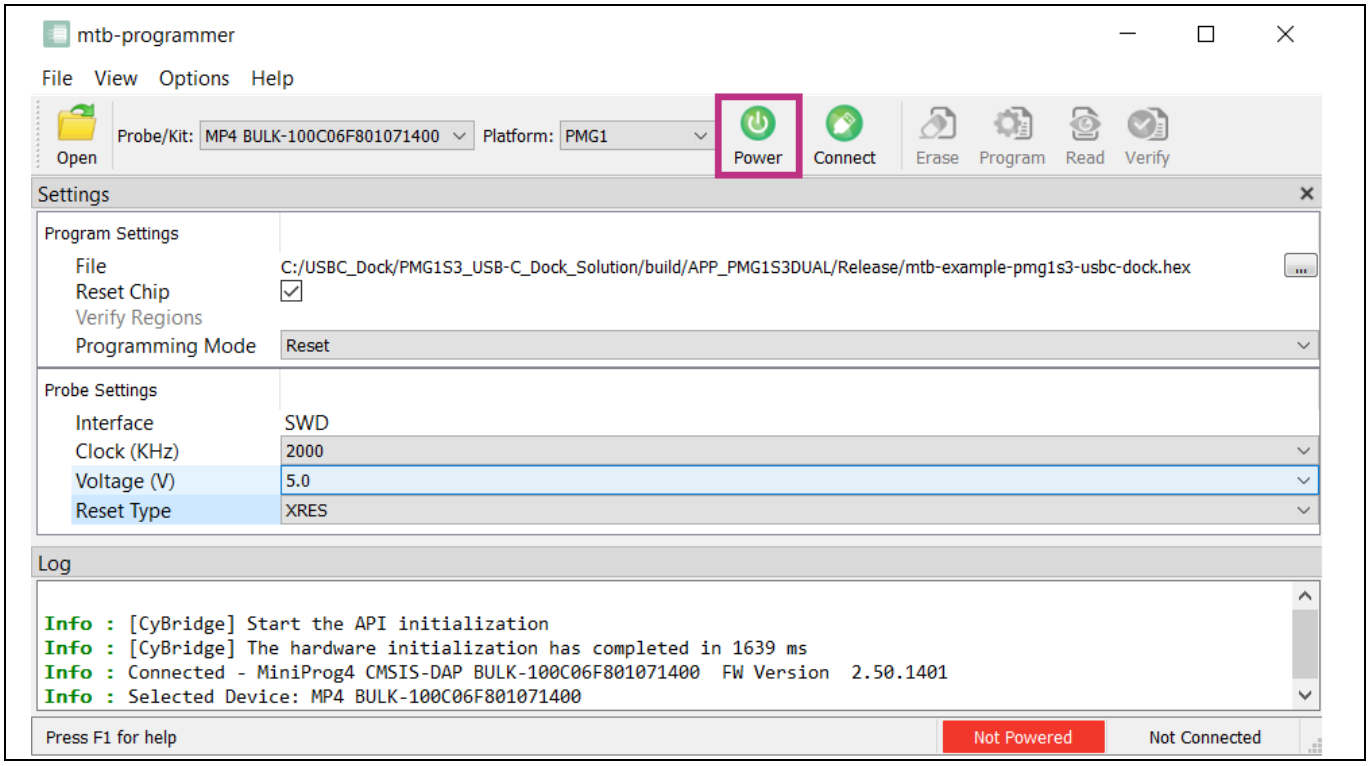

Figure 8 Powering the device

5. Click **Open**, as shown in Figure 9.

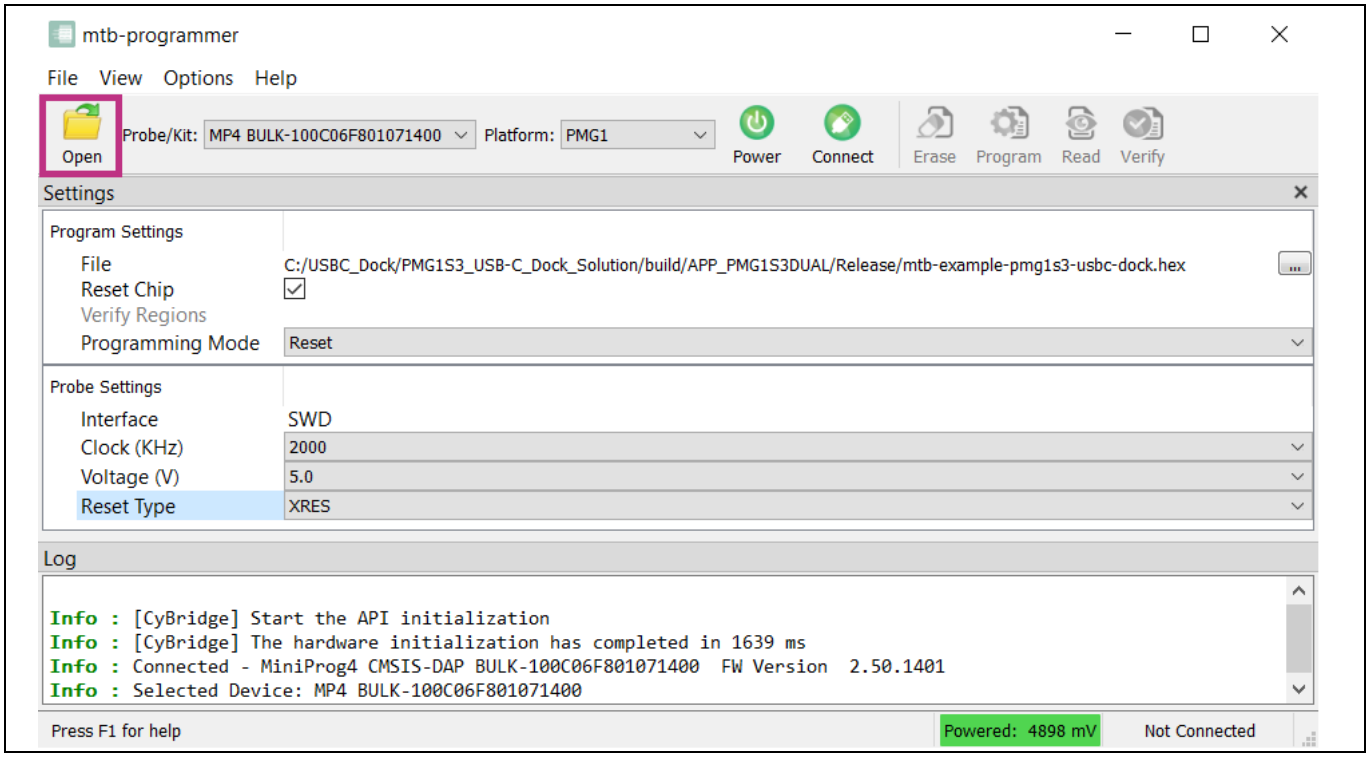

Figure 9 Opening the HEX file

The **Open Programming File** window is opened. Then, navigate to the location of the HEX file, select it, and click **Open**.

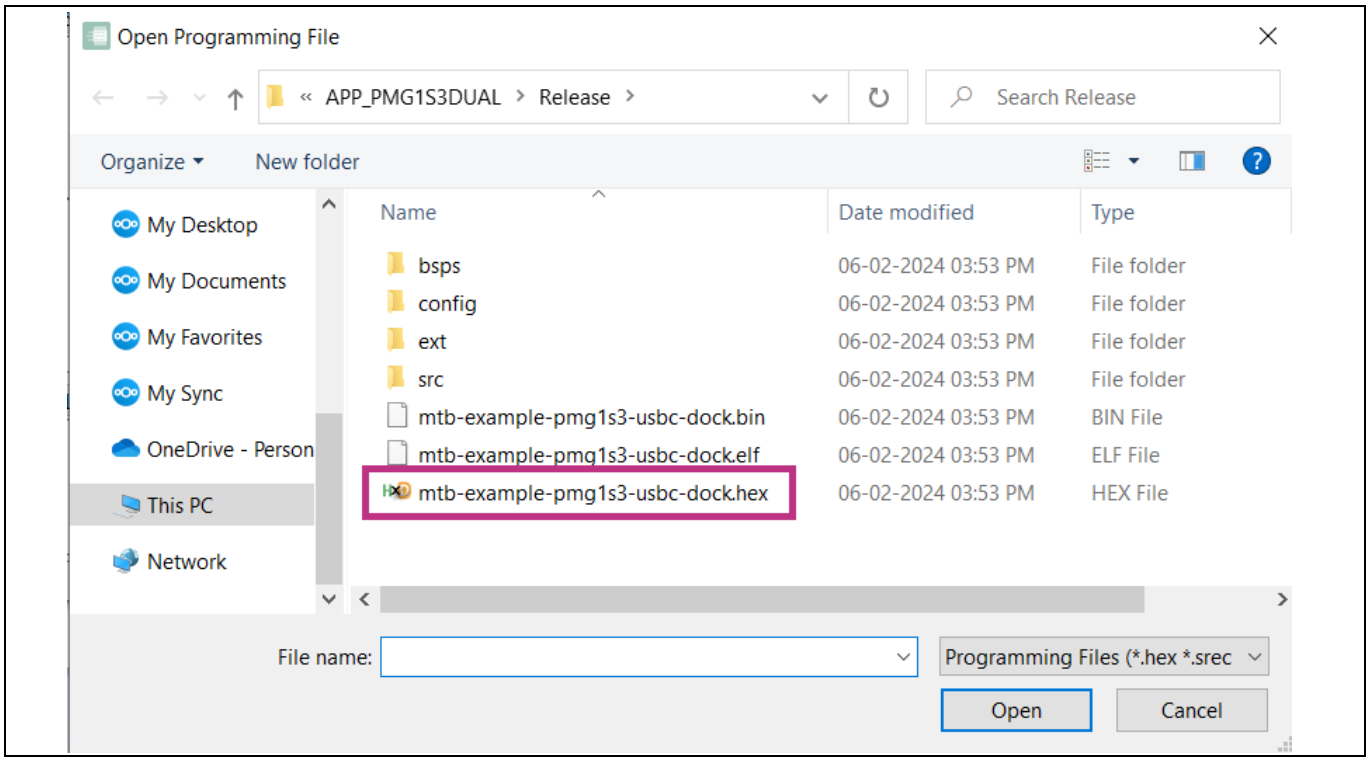

Figure 10 Selection of hex file

6. Click **Connect,** as shown in Figure 11.

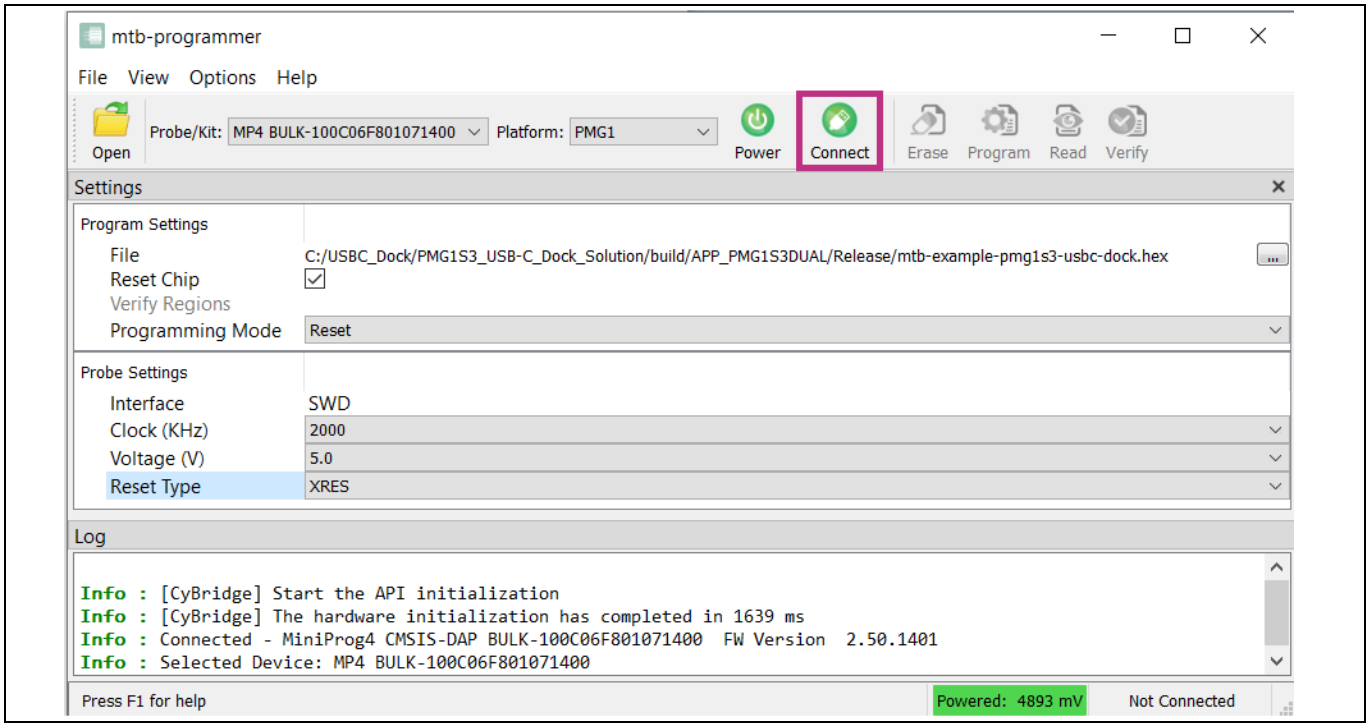

Figure 11 Connecting to target

mtb-programmer communicates with the device and displays various messages in the log. After the successful connection, the “Connected to the target device” message is displayed in the status bar.

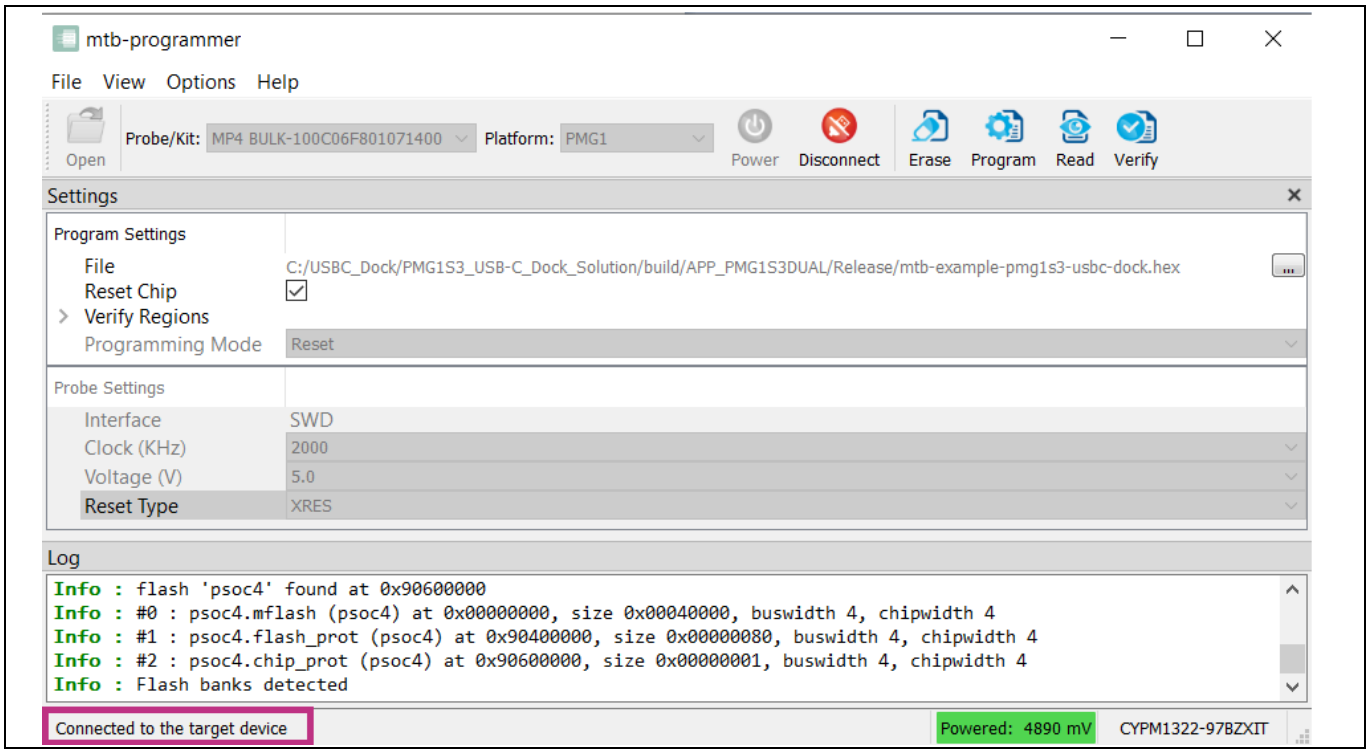

Figure 12 Indication of connection status

7. Click **Erase** to erase the flash (see Figure 13).

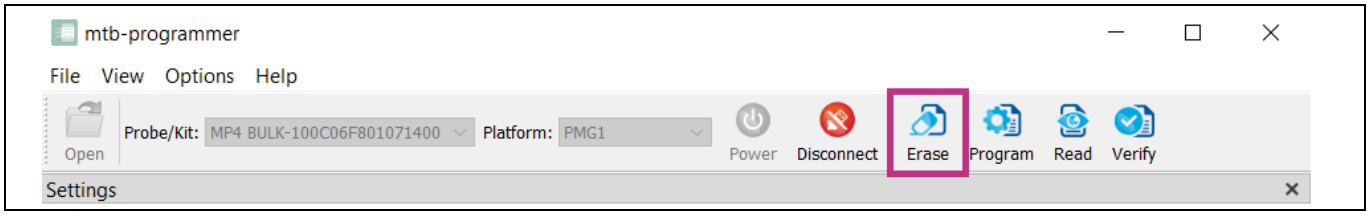

Figure 13 Erase the flash of target

When the erase operation is completed, the mtb-programmer displays the "**Device erased successfully**" message in the status bar.

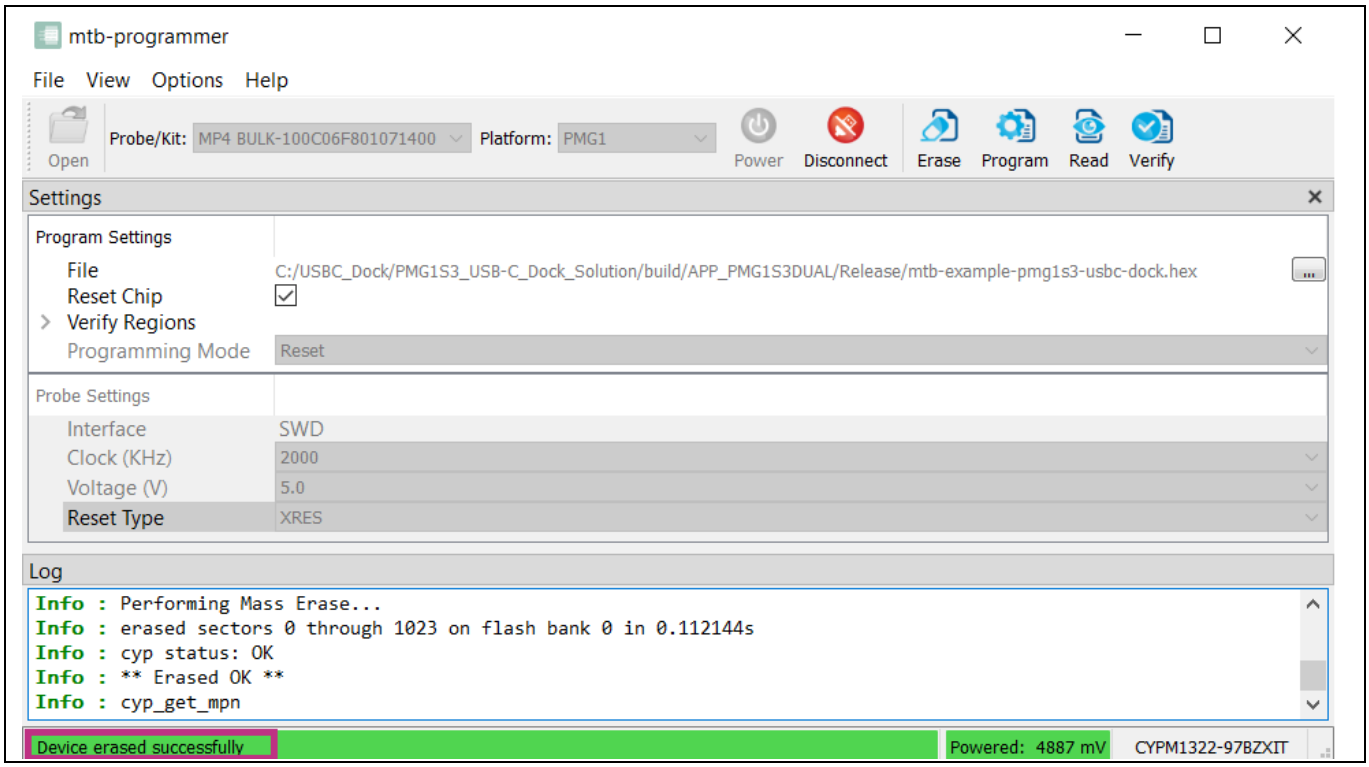

Figure 14 Indication of successful erase operation

8. Click **Program** to load the selected HEX file (see Figure 15).

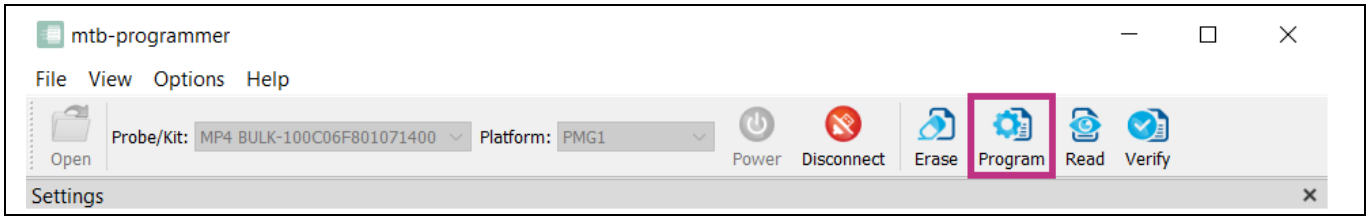

Figure 15 Programming the target

The mtb-programmer will program the target with the selected HEX file. Upon successful completion of the program operation, the “Device programmed succefully” message is displayed in the status bar.

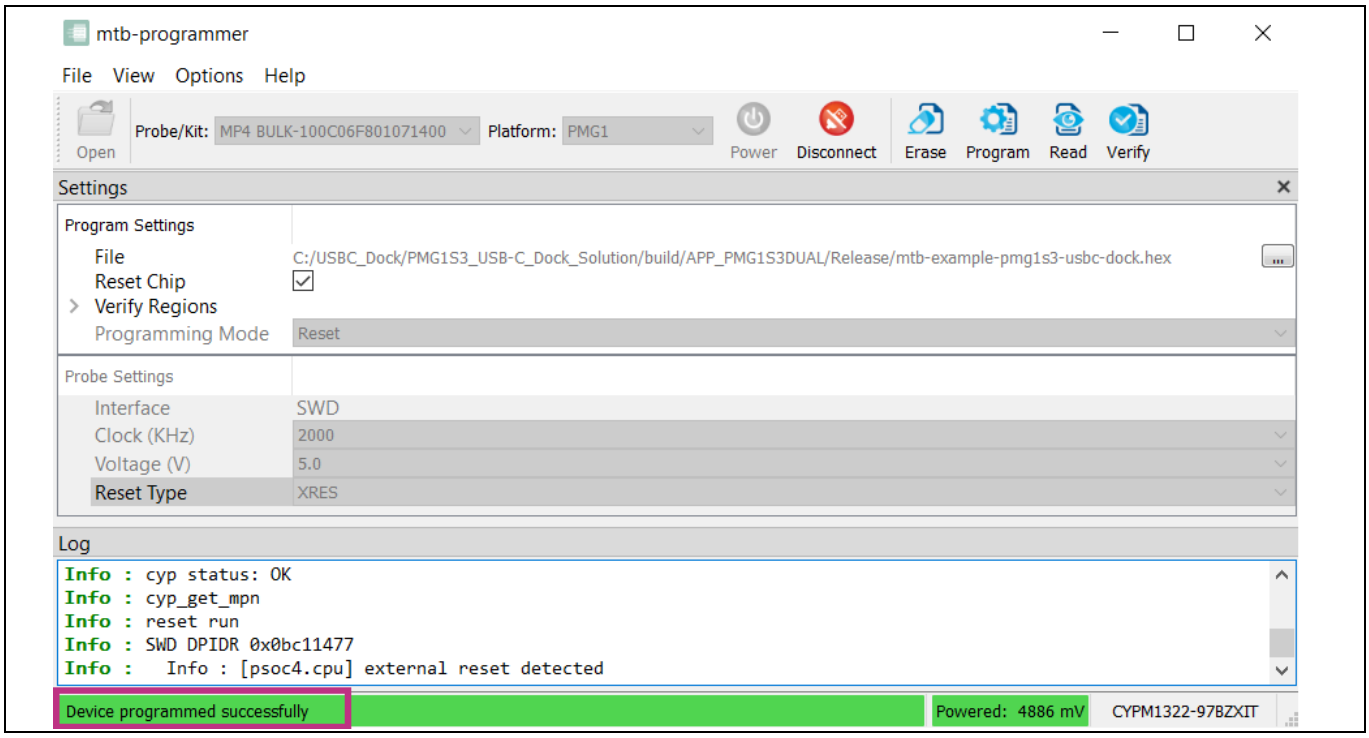

Figure 16 Indication of successful program operation

9. Click **Disconnect**, as shown in Figure 17.

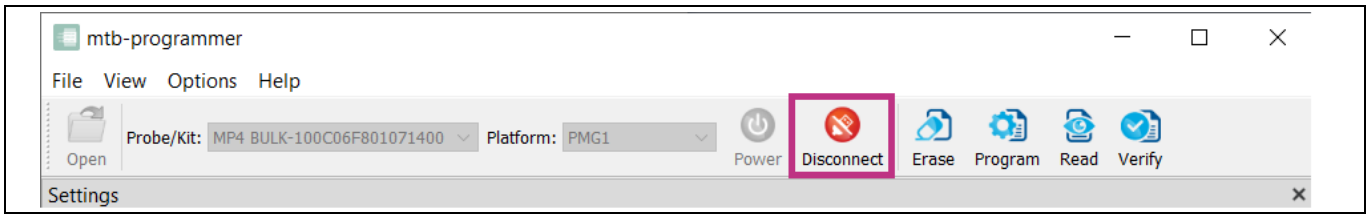

Figure 17 Disconnecting from target after programming

3.2.3.2 Using Eclipse IDE for ModusToolbox™ software

1. Connect the MiniProg4 to the programming header on the board.
2. Ensure that the device is powered. To power up the device, follow the steps 1 through 4 in the Using mtb-programmer section.
3. Select the application project in the **Project Explorer** (see Figure 18).
4. In the **Quick Panel**, under **Launches**, click **<Application Name> Program (KitProg3_MiniProg4)** (see Figure 18).

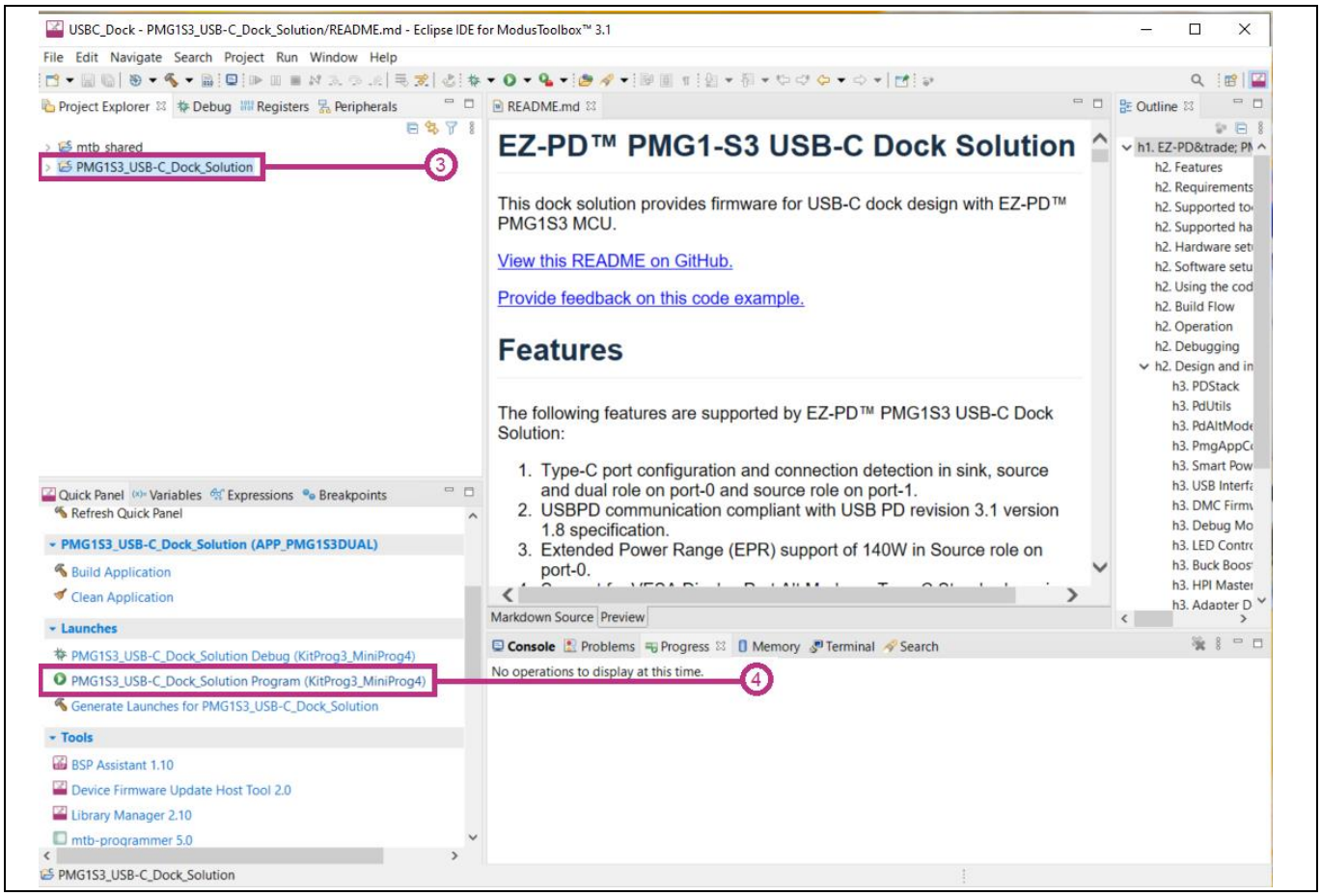

Figure 18 Programming the device using Eclipse IDE for ModusToolbox™ software

3.2.3.3 Using CLI

Follow these steps to program the application using CLI:

1. Connect the MiniProg4 to the programming header on the board.
2. Ensure that the device is powered. To power up the device, follow the steps 1 through 4 in the Using mtb-programmer section.
3. From the terminal, execute the `make program` command to build and program the application to PMG1-S3.

3.2.4 Debugging using SWD

Connect the MiniProg4 to the programming header on the board and ensure that the device is powered. To power up the device, follow the steps 1 through 4 in the Using mtb-programmer section. If the device is already flashed with a firmware, make sure that it is erased. To erase the device, follow the steps 1 through 7 in the Using mtb-programmer section.

Follow these steps to debug the application using SWD:

1. In the application Makefile, set the **CONFIG** parameter to `Debug`
2. In the application Makefile, set the **APPNAME_EXT** parameter to `fw1` (for debugging backup image) or `fw2` (for debugging primary image).
3. In the Eclipse IDE, click the **Generate Launches for <Application Name>** link in the **Quick Panel**. Wait for the operation to be completed.
4. Click **<Application Name> Debug (KitProg3_MiniProg4)** configuration in the **Quick Panel**. Ensure that the board is connected to the PC using the USB cable through the MiniProg4. See the “Debug mode” section in the kit user guide.

For example, to debug the fw1 for EZ-PD™ PMG1-S3 USB-C Dock Solution, see the steps provided in Figure 19.

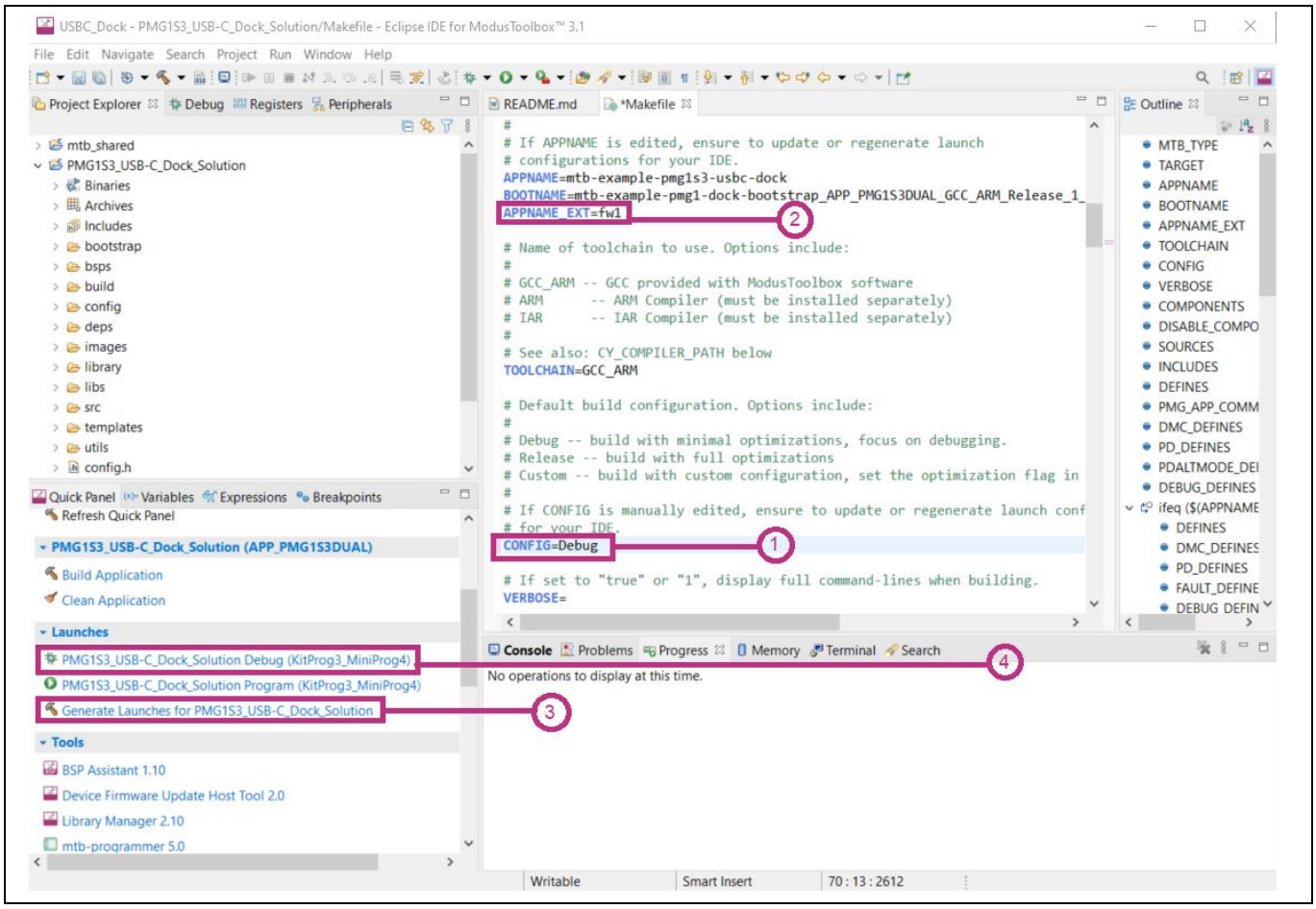

Figure 19 Debugging the application using SWD

For more details, see the "Program and debug" section in the Eclipse IDE for ModusToolbox™ software user guide.

3.3 Updating the configurations for PMG1-S3 Dock

EZ-PD™ PMG1-S3 Dock solution uses a configuration table through which the Dock solution can be customized without the need for recompiling the project.

EZ-PD™ Dock Configuration Utility enables the customers to customize several parameters of the dock.

The tool could be used for creating a new configuration from scratch, customizing the configuration by reading an existing hex file.

The tool can also be used for creating composite binaries to perform FW updates to the dock components such as PMG1-S3, CCGx devices, Intel's Goshen Ridge controller, and so on.

The configurable parameters of the dock are described in the table below. For more information on customizations, see EZ-PD™ PMG1-S3 Dock user guide.

3.3.1 Dock configuration parameters

Table 3 DMC configuration parameters

Reference	Parameter	Existing value	Description
Device parameters	Device family	PMG1	All parameters under Device parameters are read only
	Part number	CYPM1322-97BZXIT	
	Device type	DMC	
Table parameters	Table type	Dock	This is read only parameter
	Major version	6	DO NOT modify these values
	Minor version	0	
	Patch version	0	
CDTT Parameters	Dock information	–	These parameters are covered in Table 4
	Devices	–	These parameters are covered in Table 5
Public key	Public key length enable	No	To enable the signed FW update, change this field to 'yes'
	File path	Empty	Provide the public key file path
User parameters	Parameter [8:1]	0x00000000	32-byte user space

Table 4 CDTT parameters

Reference	Parameter	Existing value	Description
Dock information	Signature	C	These are read only parameters
	FWCT version	4	
	CDTT version	1	
	Vendor ID (0x)	04B4	VID value for the Dock
	Product ID (0x)	F504	PID value for the Dock
	Device ID	1	–
	Vendor name	Infineon Technologies	32-byte string indicating vendor name for the Dock. This is displayed when Dock status is queried using the EZ-PD™ Firmware Update Utility.
	Product name	PMG1S3 USBC Dock	32-byte string indicating product name for the Dock. This is displayed when dock status is queried using the EZ-PD™ Firmware Update Utility.
	Device count	3	Populated based on number of devices added under the Devices option
	Dock type	1	Read only parameter
	Firmware update mode	Unsigned (0)	You can choose between Unsigned and Signed Firmware update modes

Table 5 CDTT devices parameters

Parameter	Existing value (PMG1-S3)	Existing value (SPI flash)	Existing value (CCG7SC)	Description
Device type	240	255	241	This is a 1-byte field which is be used by PMG1-S3 to uniquely identify each firmware updatable device by PMG1-S3. The FW Update utility displays the device names based on these values. Customers should contact Infineon for new device support.
Component ID	0	1	2	Component ids are unique IDs assigned to the various onboard components of Dock. Maximum devices supported is '8'.
Image Mode	2	0	0	0: single image. The device has one application image 1: Dual image symmetric. The device has two identical images 2: Dual image asymmetric. The device has two images of different sizes
Row size (*64)	4	4	4	The row size is represented in multiples of 64 bytes
Parameter1 (0x)	0	0	8	For PMG1-S3 and SPI flash, these values are unused. For CCGx devices, which are connected on HPI Bus, the parameter1 indicates the Slave address, parameter2 indicates the GPIO port and pin number being used on PMG1-S3 side for receiving HPI interrupts from CCGx devices. Remaining parameters are unused. See the `cy_stc_app_dmc_dev_access_param_t` structure in PmgAppCommon API guide.
Parameter2 (0x)	0	0	24	
Parameter3 (0x)	0	0	0	
Parameter4 (0x)	0	0	0	
Parameter5 (0x)	0	0	0	
Parameter6 (0x)	0	0	0	
Parameter7 (0x)	0	0	0	
Parameter8 (0x)	0	0	0	

3.3.2 PD port configuration parameters

Table 6 Port 0 and Port 1 parameters

Parameter name	Description
Port information	See Table 7
PDO	See Table 8
Extended Power range	See Table 9
SCEDB configuration	See Table 10
SKEDB configuration	See Table 10
Application configuration	See **Error! Reference source not found.**
Base alternate modes	See Table 12
Discover identity	See Table 13
SVID configuration	See Table 14
Thunderbolt host configuration	See Table 15
DP mode parameters	See Table 16
Power protections	See Table 17, Table 18, Table 19, Table 20, and Table 21

Table 7 Port 0 and Port 1 information parameters

Parameter name	Default value (Port 0)	Default value (Port 1)	Description
Signature (0x)	50445343	50445343	Signature being used for validating configuration table. This field is read only.
Version (0x)	0100	0100	Version of this port configuration table. This field is read only
Manufacturer vendor ID (0x)	04B4	04B4	Manufacturer vendor and product IDs. This information along with manufacturer name is used in response to PD Get_Manufacturer_Info Message.
Manufacturer product ID (0x)	F504	F504	
Manufacturer name	Infineon	Infineon	Manufacturer string being reported as part of Manufacturer Info Message. Maximum length is 22-bytes.
Port role	Dual Role	Source	Sink: The PD port can act as power sink only Source: The PD port can act as power source only Dual role: The PD port can act as power sink and power source
Default port role	Source	Source	Default role of the PD port

Parameter name	Default value (Port 0)	Default value (Port 1)	Description
Current level	3A	3A	Rp value that is being presented by the port partner: • 900 mA • 1.5 A • 3 A
Cable discovery count (0x)	14	14	Number of cable discovery attempts made
Rp-Rd toggle	Yes	No	Whether Rp-Rd toggle is allowed or not. This is enabled only when the port role is dual role.
Rp supported	900 mA 1.5 A 3 A	None	Rp values supported by the device. This field supports eight values. • None • 900 mA • 1.5 A • 3 A • 900 mA, 1.5 A • 900 mA, 3 A • 1.5 A, 3 A • 900 mA, 1.5 A, 3 A
PD operation enable	Yes	Yes	Whether USB PD operation is supported on the port
Preferred power role	Try.Sink Supported	No Try.Src or Try.Sink supported	See USB Type-C specification regarding these parameters.
Port disable	No	No	Whether the PD port is to be disabled at start-up. Can be enabled through API call afterwards.
Cable discovery enable	Yes	Yes	Whether cable discovery is enabled or not.
Dead battery support enable	Yes	Yes	Not applicable with PMG1-S3 Dock SDK or self-powered Docks
Error recovery enable	Yes	Yes	Whether the Type-C error recovery is enabled or not
Accessory mode enable	Yes	Yes	Accessory mode enable/disable parameter
Rp detach enable	Yes	Yes	Option to enable/disable disconnect detect mechanism using Rp in sink role
Vconn retain	No	No	Whether the VCONN supply should be left enabled even if the EMCA cable indicates that VCONN is not required

Parameter name	Default value (Port 0)	Default value (Port 1)	Description
FRS configuration	None	None	This parameter specifies the fast role swap configuration for the given port. This solution does not support fast role swap.
Port type	Shared capacity port	Assured capacity port	See the latest USB PD specification for information related to these parameters. These parameters are used to form the Source_Info Message.
Port maximum PDP (W)	140	15	
Port present PDP (W)	140	15	
Port reported PDP (W)	140	15	
Revision major	3	3	The USB PD specification revision and version with which the FW is complaint.
Revision minor	1	1	
Version major	1	1	
Version minor	8	8	

Table 8 Port 0 and Por 1 SPR PDO

Parameter name	Default value (Port 0)	Default value (Port 1)	Description
Port-0 source PDO's [6:0]	0x2781912C	0501912C	See the USB PD specification for more details
–	0x0002D12C	NA	
–	0x0004B12C		
–	0x000641F4		
–	0xC9A43264		
Port-0 sink PDO [6:0]	0x26019000	NA	

Table 9 Port 0 EPR PDO's

Parameter name	Default value	Description
EPR source PDO [4:0]	0x0008C1F4	See the USB PD specification for more details
–	0xD230968C	

Table 10 Port 0 and Port 1 SCEDB and SKEDB

Parameter name	Description
SCEDB	See the source capabilities extended message in latest USB PD specification for SCEDB parameter
SKEDB	See the sink capabilities extended message in latest USB PD specification for SKEDB parameter

Table 11 Port 0 and Port 1 application configuration

Parameter name	Default value (Port 0)	Default value (Port 1)	Description
Preferred data role	UFP	DFP	See the latest USB PD specification about UFP and DFP
Preferred power role	Sink	Source	Port 0's preferred role is sink Port 1's preferred power role is source
USB3 enabled	Unused	Unused	Application specific alternate mode parameters. These are unused.
MFDP enabled			
TBT3 enabled			
USB4 enabled			
DP 4 lane enabled			
Sink USB suspend enable	No	No	Whether the sink supports USB suspend
DR SWAP response	ACCEPT	REJECT	See the USB PD specification for more details on these commands
PR SWAP response	ACCEPT	REJECT	
VCONN SWAP response	ACCEPT	ACCEPT	

Table 12 Port 0 and Port 1 base alternate modes

Parameter name	Sub parameter	Default value (Port 0)	Default value (Port 1)	Description
Alternate mode 0	SVID# 0 (0x)	FF01	FF01	Alternate mode SVID value supported by Port 0or Port 1. Customers add up to 4 SVID values per alternate mode that can coexist simultaneously.
–	Supported in UFP	Yes	No	Indicates if this SVID is to be supported when the PD port is UFP.
–	Supported in DFP	No	Yes	Indicates if this SVID is to be supported when the PD Port is DFP.
Alternate mode [3:1]	–	–	–	Customers can add 3 additional Alternate modes [3:1]. Customers can add up to 4 SVID values per alternate mode which can coexist simultaneously.

Table 13 Port 0 and Port 1 discover identity device IDs

Parameter name	Sub parameter	Default value (Port 0)	Default value (Port 1)	Description
Device IDs	USB host support	No	Yes	See the "**Discover Identity**" section in the USB PD specification.
	USB device support	Yes	No	
	Modal operation supported	Yes	Yes	
	USB vendor ID	0x04B4	0x04B4	
	Product type (UFP)	Hub	Undefined	
	Product type (DFP)	Undefined (Not applicable for US Port)	Hub	
	Connector type	USB Type-C Receptacle	USB Type-C receptacle	
	USB-IF compliance XID	0x00000000	0x00000000	
	USB product id	0xF504	0xF504	
	BCD device	0x0000	0x0000	
UFP VDO	UFP VDO Version	V1.3	–	See the "**UFP VDO"** in the latest USB PD specification.
	USB 2.0 device capability	Capable	–	
	USB 3.2 device capability	Not capable	–	
	USB 4 device capability	Not capable	–	
	VConn Power(W)	1	–	
	VConn required	No	–	
	VBUS required	Yes	–	
	TBT3 mode enable	No	–	
	Other data mode enable	Yes	–	
	Custom mode enable	Yes	–	
	USB highest speed	USB3.2 Gen1	–	

Table 14 Port 0 SVID configuration

Parameter name	Existing value	Description
SVID Value	FF01	SVID value of the Alternate mode
Mode 0	0x1C0045	DP SID Capabilities, which is reported for USB PD Discover modes.

Table 15 Port 0 and Port 1 Thunderbolt host configuration: applicable for Griffin Creek Dock only

Parameter name	Default value (Port 0)	Default value (Port 1)	Description
Enable	Yes	Yes	This is a Boolean option. Unused by SDK
Thunderbolt controller	Unused by SDK	Unused by SDK	–
HPD handling	Virtual HPD	Virtual HPD	GPIO based HPD for TBT based dock is not supported
VPRO capable	Yes	Yes	VPRO enabled Dock
SBU MUX configuration	Not applicable	Not Applicable	SBU Mux configuration to be used
USB4 data role	Not applicable	Host	This option is applicable for Port 1 only 1. None 2. Host 3. Dual role 4. Device
USB3 data role	Not applicable	Host	This option is applicable for Port 1 only 1. None 2. Host 3. Dual role 4. Device
USB4 host support	Not applicable	Enable	Enable/disable USB4 host support
Support TBT tunneling		Yes	These options are enabled when USB4 host support is enabled. Applicable only for Port 1.
Support DP tunneling		Yes	
Support PCIe tunneling		Yes	
Non-thunderbolt mux	No	No	This parameter is not used in the current SDK

Table 16 Port 0 and Port 1 DP mode parameters for USB-C Dock and Griffin Creek Dock

Parameter name	Default value (Port 0)	Default value (Port 1)	Description
Pin A configuration supported	False	False	Refer VESA DisplayPort Alt mode on USB Type-C standard
Pin B configuration supported	False	False	Refer VESA DisplayPort Alt mode on USB Type-C standard
Pin C configuration supported	True	True	Refer VESA DisplayPort Alt mode on USB Type-C standard
Pin D configuration supported	True	True	Refer VESA DisplayPort mode on USB Type-C standard
Pin E configuration supported	True	True	Refer VESA DisplayPort Alt mode on USB Type-C standard
Pin F configuration supported	False	False	Refer VESA DisplayPort Alt mode on USB Type-C standard
Mux control	Controlled by CCGx	Controlled by CCGx	This parameter is not used in this SDK
Mode trigger	Not Applicable	Automatic	For dock applications this parameter is set to Automatic for Port 1
Operation	DP sink	DP source	This parameter is not used in this SDK
Preferred DP mode	4 lanes display port	4 lanes display port	This parameter is not used in this SDK

Table 17 Port 0 and Port 1 overvoltage protection

Parameter name	Default value (Port 0)	Default value (Port 1)	Description
Enable	Yes	Yes	Whether to enable/disable OVP
VBUS OVP mode selection	OVP using dedicated comparator, hardware detects trips and turns off FETs	OVP using dedicated comparator, hardware detects trips and turns off FETs	VBUS OVP mode selection 0 - OVP using ADC comparator 1 - OVP using dedicated comparator, Firmware detects trips and turns off FETs 2 - OVP using dedicated comparator, Hardware detects trips and turns off FETs *Note: Only Mode 2 is supported in this SDK*
OVP threshold (%)	20	20	OVP threshold: Percentage of Excess voltage above contract voltage when OVP handling should start
Debounce duration (us)	10	10	OVP debounce duration in microseconds
Retry count	2	2	Number of consecutive OVP events allowed before the port operation is suspended by the firmware

Table 18 Port 0 and Port 1 overcurrent protection

Parameter name	Default value (Port 0)	Default value (Port 1)	Description
OCP enable	Yes	Yes	Whether to enable/disable OCP
VBUS OCP mode selection	Internal OCP with neither software debounce nor automatic FET control	Internal OCP with neither software debounce nor automatic FET control	VBUS OCP mode selection 0 - OCP using external hardware 1 - Internal OCP with neither software debounce nor automatic FET control 2 - Internal OCP with automatic FET control by hardware when an OCP event is detected 3 - Internal OCP with software debounce using delay in milliseconds specified by the user
OCP threshold (%)	20	20	Excess current in percentage of maximum allowed current

Parameter name	Default value (Port 0)	Default value (Port 1)	Description
Debounce duration (ms)	10	10	OCP debounce period in milliseconds. The OCP handling is only performed if the OCP condition persists for this duration.
Retry count	2	2	Number of consecutive OCP events allowed before the port is suspended by the firmware
Sense resistance (milli-ohm)	5	5	The sense resistor impedance in milli-ohm unit
OCP threshold 2 (%)	0	0	Secondary OCP threshold (corresponding to peak current condition). Set to '0' if not used.
Debounce duration 2 (ms)	0	0	Debounce period in ms corresponding to secondary threshold
Tuning resistance (Ω)	0	0	Current sense tuning resistor impedance in 100 Ω unit

Table 19 Port 0 and Port 1 undervoltage protection

Parameter name	Default value (Port 0)	Default value (Port 1)	Description
Enable	Yes	Yes	Whether to enable/disable UVP
VBUS UVP mode selection	UVP using dedicated comparator, hardware detects trips and turns off FETs	UVP using dedicated comparator, hardware detects trips and turns off FETs	VBUS UVP mode selection 0 - UVP using ADC comparator 1 - UVP using dedicated comparator, Firmware detects trips and turns off FETs 2 - UVP using dedicated comparator, Hardware detects trips and turns off FETs *Note: Only Mode 2 is supported in this SDK*
UVP threshold (%)	70	70	Reduced voltage below expected value in percentage of expected voltage
Debounce duration (µs)	1	1	UVP debounce duration in microseconds. If a non-zero debounce is specified, there can be an error of up to 35 µs due to the device being in Sleep mode.
Retry count	2	2	Number of consecutive UVP events allowed before the port operation is suspended by the PD firmware.

Table 20 Port-0 and Port-1 short-circuit protection

Parameter name	Default value (Port 0)	Default value (Port 1)	Description
Enable	Yes	Yes	Whether to enable/disable SCP
SCP threshold (%)	50	50	SCP threshold: Excess current in percentage of maximum allowed current. This is not used in the current SDK
Debounce duration	1	1	SCP debounce duration in microseconds. This should ideally be set to '0'.
Retry count	2	2	Number of consecutive SCP events allowed before the port operation is suspended
Sense resistance (milli-ohm)	5	5	Sense resistor impedance in milli-ohm unit

Table 21 Port-0 and Port-1 Vconn OCP

Parameter name	Default value (Port 0)	Default value ([Port 1)	Description
Enable	Yes	Yes	Whether to enable/disable VCONN OCP
Threshold (%)	30	30	Maximum Vconn current allowed in 10 mA unit
Debounce duration (mS)	1	1	VCONN OCP debounce period in milliseconds
Retry count	0	0	Number of consecutive OCP events allowed before the port is suspended by PD firmware.

3.3.3 Description of smart power parameters

Table 22 Smart Power parameter

Parameter name	Default value	Description
Smart power support	Yes	Users can enable or disable Smart power using this option. When Smart power is disabled fixed PDO's are advertised over upstream port (US) and no power throttling is performed. This is a Boolean value.
Adaptor detection	Yes	Whether adaptor detection to be performed by FW or not. This is a Boolean value.
Adaptor power	120	When adaptor detection is not supported by system to enable Smart power users could provide the Adaptor size in watts using this parameter. Value is represented in decimal and maximum value is 280 W.
Buffer power	5	The power to be reserved for additional usage by onboard components. Maximum value is 10 W.
Step size	5	The step size used to incrementally update the power advertised on US. The value is represented in decimal and maximum value allowed is 5W.
Interval of power monitoring	2000	This period specifies the periodicity with which the firmware monitors the power consumption. The value is represented in decimal and units are milliseconds and maximum value is 10000 ms.
Debounce count	10	Debounce period before incrementing the US power when DS ports stop consuming the power. The value is represented in decimal and maximum value is 20.
Max upstream power	140	Maximum US power allowed to be advertised when smart power is enabled. The value is represented in decimal and maximum allowed value is 140.

3.3.4 Description of Billboard (BB) parameters

Table 23 Billboard (BB) parameters

Parameter name	Default value	Description
BB version	0122	BB version supported by the SDK. PMG1-S3 Dock SDK is compliant with BB v1.2.2
BB type selection	Internal billboard	Type of billboard implementation. This SDK support only internal billboard.
BB operation control	Enable BB USB always	The billboard interface is always enumerated in the Dock solution. Do not modify this parameter.
USB interface selection	HID interface enabled	This is not used in the current SDK
BB timeout (mS)	0	This is not used in the current SDK
Current drawn (mA)	50	This value is used by PMG1-S3 as part of Configuration Descriptor -> MaxPower. The user needs to input this value in Hex and the value is in multiples of 2 mA.
BB Bus powered enable	No	It indicates if PMG1-S3 is self-powered or bus-powered USB device
Vconn Power	32768	VCONN power needed by the device. This is a 2-byte bitmap and bit 15 needs to be set to indicate that the device does not need Vconn power. This value is used to populate VCONN power field (offset 6) of Billboard capability descriptor.
Preferred alternate mode	00	This refers to the field PreferredAlternateOrUSB4Mode (Offset 5) of Billboard capability descriptor
Unique container ID	00	Not used in current SDK
BB device vid	04B4	The hex value used as VID when device enumerates. Customers should modify this with their USB-IF assigned vendor ID.
BB device pid	F504	The hex value being used as USB PID when device enumerates.
BB manufacturer string	Infineon Technologies	Maximum allowed length is 31-bytes
BB product string	DMC device	Maximum allowed length is 31-bytes

Parameter name	Default value	Description
BB unique serial	0	Maximum allowed length 31 bytes. A zero value here indicates that the BB serial string needs to be used in serial string descriptor. A non-zero value here indicates that the firmware should generate the serial string descriptor. *Note: Using a non-zero value here will require custom modifications to be made in the reference projects*
BB serial string	123456	Maximum allowed length is 31 bytes
BB config string	Billboard configuration	Maximum allowed length is 31 bytes
BB interface string	Billboard interface	Maximum allowed length is 31 bytes
BB hid interface string	HID interface	Maximum allowed length is 31 bytes
BB additional url string	https://www.infineon.com/Type-C	Maximum allowed length is 63 bytes
BB alternate string array	['Type-C Alternate Mode1', 'Type-C Alternate Mode2']	Maximum allowed length is 256 bytes. Maximum number of strings allowed is 8.

4 Structure of the project

After the Project creation process (see the section 3.2.1), two folders will be created for each of the reference projects included as part of this SDK. They are:

- **mtb_shared:** This folder contains the middleware libraries that provide key features to the application along with few utilities for building and programming ModusToolbox™-based applications.
- **Application:** This folder contains the header, source files and utilities that provide user configurations, user hardware-specific functions and custom code modules. It is expected that these files may be changed to match the hardware design and requirements for all customer implementations.

Table 24 lists the middlewares used by the reference projects.

Table 24 Middlewares used by reference projects

Middleware	Source/library	USB-C Dock solution	Griffin Creek Dock solution
PdStack	Library	Used	Used
PdAltmode	Source and library	Only source files are used	Both source and library files are used
PmgAppCommon	Source and library	Both source and libraries are used	Both source and libraries are used
PdUtils	Source	Used	Used
USBDEV	Source	Used	Used
mtb-pdl-cat2	Source	Used	Used
HPI	Library	Used	Not used

The header and source files, and utilities used in the application for each project are described in the following sections.

4.1 PMG1-S3 USB-C Dock solution

The structure of PMG1-S3 USB-C Dock solution is shown in Figure 20.

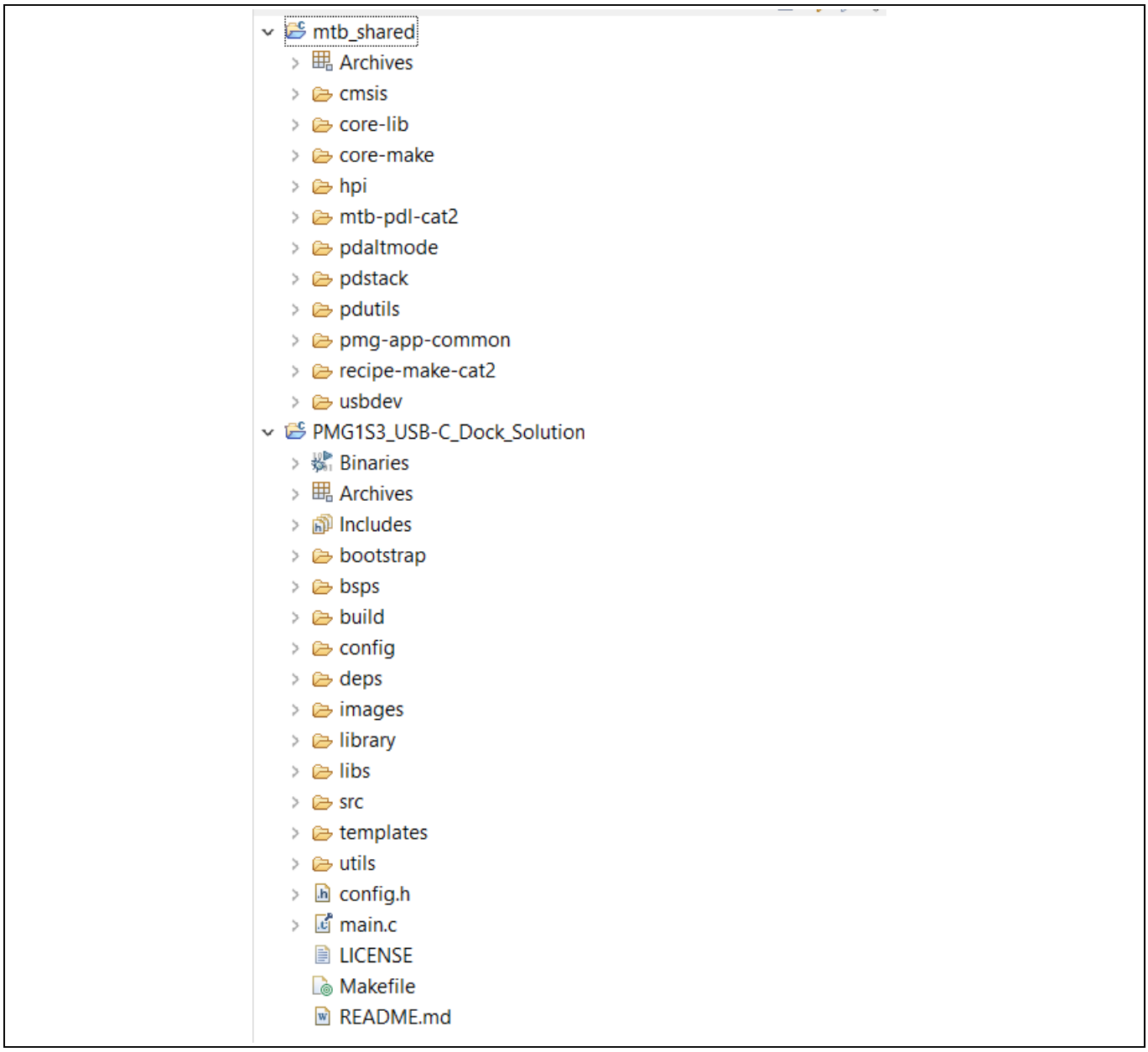

Figure 20 Structure of PMG1-S3 USB-C Dock solution

Structure of the project

Table 25 lists the application source and header files along with their purpose.

Table 25 Source and header files used in PMG1-S3 USB-C Dock solution

File	Purpose
src/custom_altmode/custom_altmode_vid.c *src/custom_altmode/custom_altmode_vid.h*	Defines data structures, function prototypes and implements functions for custom alternate mode support
src/dmc/ccgx_ctrl.c *src/dmc/ccgx_ctrl.h*	Defines function prototypes and implements functions to handle CCGx firmware update
src/dmc/cryptolite_rsa.c *src/dmc/cryptolite_rsa.h*	Defines function prototypes and implements RSA signature verification
src/dmc/dmc_flashing.c *src/dmc/dmc_flashing.h*	Defines function prototypes and implements functions for DMC (PMG1-S3) firmware update
src/dmc/dmc_solution.c *src/dmc/dmc_solution.h*	Defines function prototypes and implements functions for DMC interfaces initialization
src/dmc/etag.c *src/dmc/etag.h*	Defines data structures and function prototypes associated with the storage and retrieval of ETAG information
src/dmc/spi_eeprom_master.c *src/dmc/spi_eeprom_master.h*	Defines function prototypes and implements functions for SPI master interface to SPI EEPROM slave
src/mux/anx7443.h	Defines function prototypes for ANX7443 MUX driver
src/mux/pericom_pi3usb31532.c *src/mux/pericom_pi3usb31532.h*	Defines function prototypes and implements driver for Pericom PI3USB31532 MUX
src/app_version.h	Defines application firmware version
src/pmg1_version.h	Defines the base firmware stack version
src/solution.c *src/solution.h*	Defines function prototypes and implements solution specific functions
src/usb_descr.c *src/usb_descr.h*	Defines data structure and implements functions to update USB descriptors
src/usb_hid.c *src/usb_hid.h*	Defines data structures and function prototypes associated with the HID interface
config/config.c	Defines dock configuration table
config.h	Defines feature macros for the application configuration
main.c	Implements application main logic

The following utilities are also part of the application. These are located inside the *utils* folder:

- **bin2psochex**

  The Bin2psochex tool is used for converting a firmware binary (*.bin*) file into a PSoC™ format hex file that is acceptable by the ModusToolbox™ programmer application. The tool uses the Intel Hex file format and also includes silicon vendor defined records for different chip families.

- ***hex_bin_update.py***

  The *hex_bin_update.py* script calculates the SHA-256 hash of the firmware image from the provided hex file and then embeds it into the same hex file. This calculated hash is used later for an integrity check before loading the application image.

4.2 PMG1-S3 Griffin Creek Dock solution

The structure of PMG1-S3 Griffin Creek Dock Solution is shown in Figure 21.

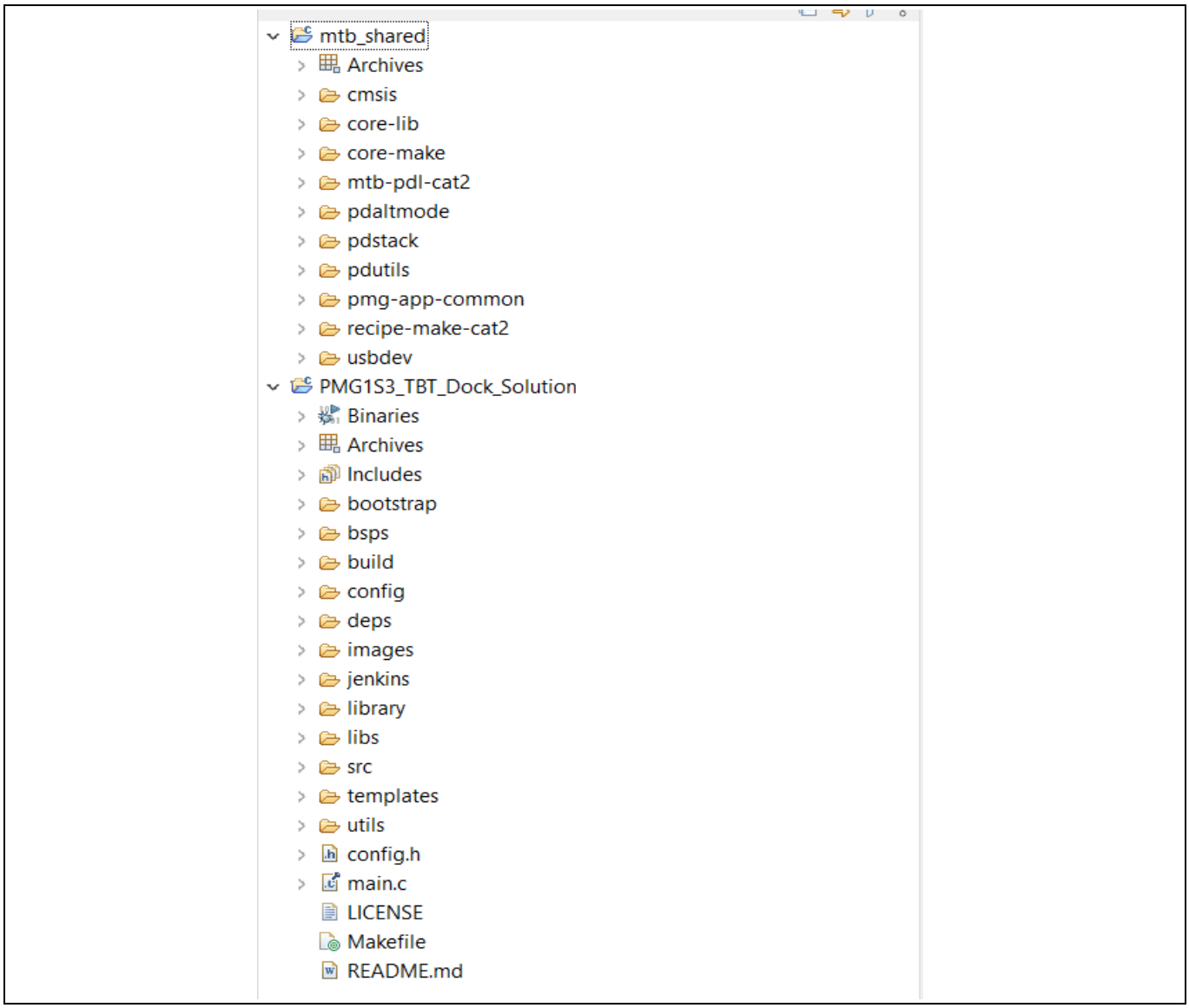

Figure 21 Structure of PMG1-S3 Griffin Creek Dock solution

Structure of the project

Table 26 lists the application source and header files along with their purpose:

Table 26 Source and header files used by PMG1-S3 Griffin Creek Dock solution

File	Purpose
src/dmc/cryptolite_rsa.c *src/dmc/cryptolite_rsa.h*	Defines function prototypes and implements RSA signature verification
src/dmc/dmc_flashing.c *src/dmc/dmc_flashing.h*	Defines function prototypes and implements functions for DMC (PMG1-S3) firmware update
src/dmc/dmc_solution.c *src/dmc/dmc_solution.h*	Defines function prototypes and implements functions for DMC interfaces initialization
src/dmc/foxville.h	Defines function prototypes for foxville firmware update
src/dmc/goshen_ctrlr_i2c_master.h	Defines function prototypes for I2C communication with GoshenRidge (GR) Controller
src/dmc/goshen_ctrlr_update.h	Defines function prototypes for Thunderbolt (TBT) Controller firmware update
src/dmc/spi_eeprom_master.c *src/dmc/spi_eeprom_master.h*	Defines function prototypes and implements functions for SPI master interface to SPI EEPROM slave
src/app_version.h	Defines application firmware version
src/fl5801.c *src/fl5801.h*	Defines function prototypes and implements functions for communication with FL5801 USB hub
src/pmg1_version.h	Defines the base firmware stack version
src/solution.c *src/solution.h*	Defines function prototypes and implements solution specific functions
src/usb_descr.c *src/usb_descr.h*	Defines data structure and implements functions to update USB descriptors
config/config.c	Defines dock configuration table
config.h	Defines feature macros for the application configuration
main.c	Implements application main logic

The following utilities are also part of the application that are located inside the *utils* folder:

- **bin2psochex**

  The Bin2psochex tool is used for converting a firmware binary (*.bin*) file into a PSoC™ format hex file that is acceptable by the ModusToolbox™ programmer application. The tool uses the Intel Hex file format and includes silicon vendor defined records for different chip families.

- ***hex_bin_update.py***

  The hex_bin_update.py script calculates the SHA-256 hash of the firmware image from the provided hex file and then embeds it into the same HEX file. This calculated hash is used later for an integrity check before loading the application image.

5 Firmware architecture

EZ-PD™ PMG1-S3 has a 256 KB flash memory that is designated to store a bootloader, along with two copies of the firmware binary along with the corresponding configuration table. The flash memory map for the device is shown in Figure 22.

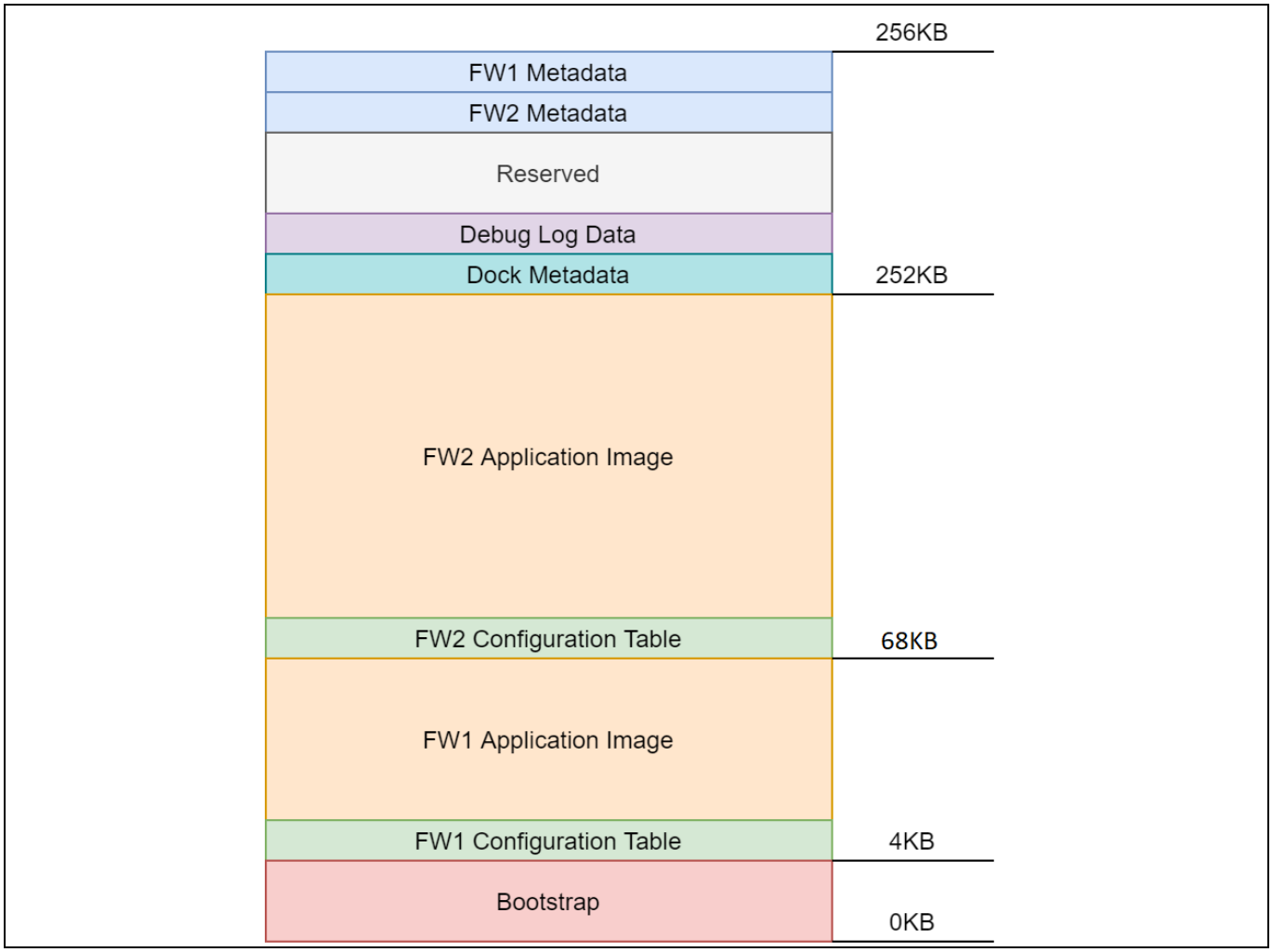

Figure 22 PMG1-S3 flash memory map

The same memory map is used for both the reference projects that are provided along with PMG1-S3 Dock SDK.

The bootstrap is located at the beginning of the flash and is responsible for checking the integrity of Firmware 1 and Firmware 2 images of PMG1-S3. The integrity check involves computing the SHA-256-bit hash of the images and comparing it against the hash stored in the firmware image's respective metadata area.

The firmware 1 (FW1) also known as backup firmware is a reduced feature firmware image that supports Phase-2 update of firmware update process: updating primary firmware image of PMG1-S3 from SPI Flash.

The firmware 2 (FW2) also known as primary firmware is a full features firmware image that supports all Dock functionality including the PD and DMC features.

5.1 Solution architecture

The Dock applications allow a connected Notebook to charge while sending out display port data and enabling connectivity to the USB HUB present within the dock. PMG1-S3 acts as a DMC controller and supports Type-C USB-PD ports and it provides the connectivity to other devices present on the Dock board.

PMG1-S3 supports the firmware update of itself and the connected component on the dock such as Goshen Ridge, Foxville controller, and CCG7SC device.

Figure 23 shows the high-level solution architecture of the PMG1-S3 Dock firmware:

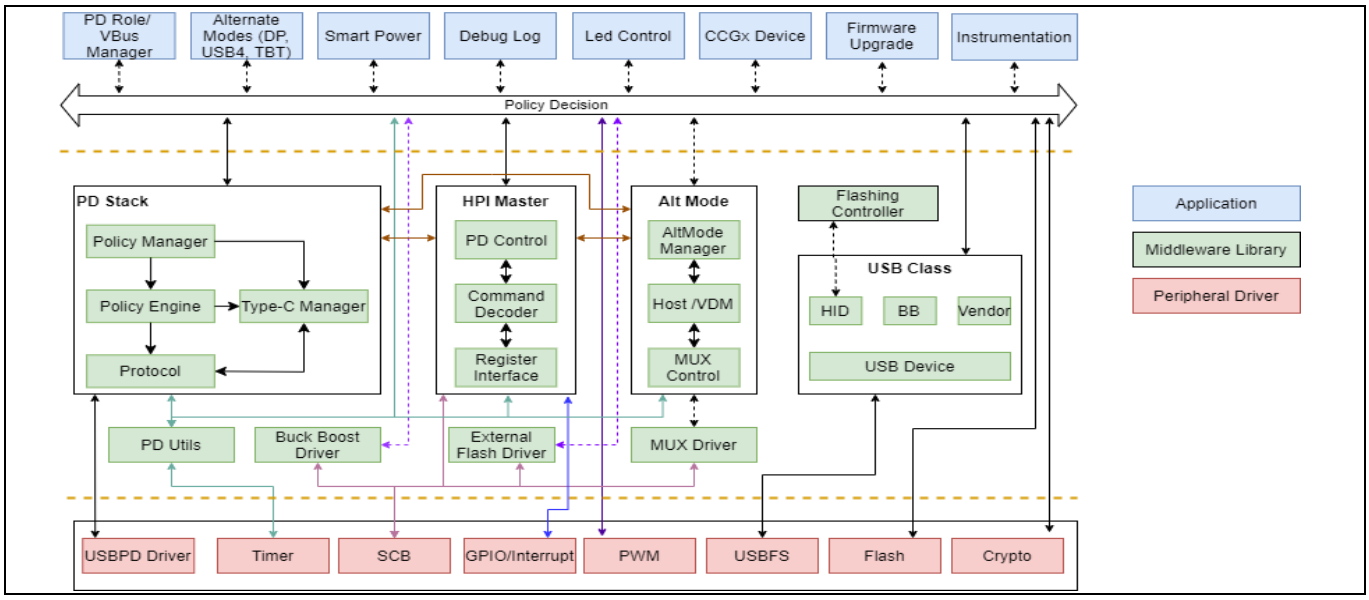

Figure 23 High-level solution architecture of PMG1-S3 dock firmware

See the Firmware operation section for a detailed understanding of firmware flow. The Firmware blocks section describes the modules used in the PMG1-S3 Dock firmware.

5.2 Firmware blocks

5.2.1 Dock application firmware

5.2.1.1 Adapter detection

Adapter detection module monitors the connection of barrel adapter and determines the power rating of the adapter using different methods. These methods are:

- **Dynamic**: The power rating of the adapter can possibly be read from the adapter itself provided the adapter supports such feature. Since this might vary from one hardware to another, this method needs to be implemented by the user depending on the adapter used in the design.
- **Static**: The power rating information is configurable through the configuration space parameter. This method is used in PMG1-S3 Griffin Creek Dock solution.
- **Jumper configuration**: On board jumper combinations can be used to determine the power rating of supported adapters. This method is only applicable on the EZ-PD™ PMG1-S3 USB-C Dock board. Figure 24 shows the combination of jumper with resistor divisor circuit to determine the power rating of adapter in EZ-PD™ PMG1-S3 USB-C Dock board.

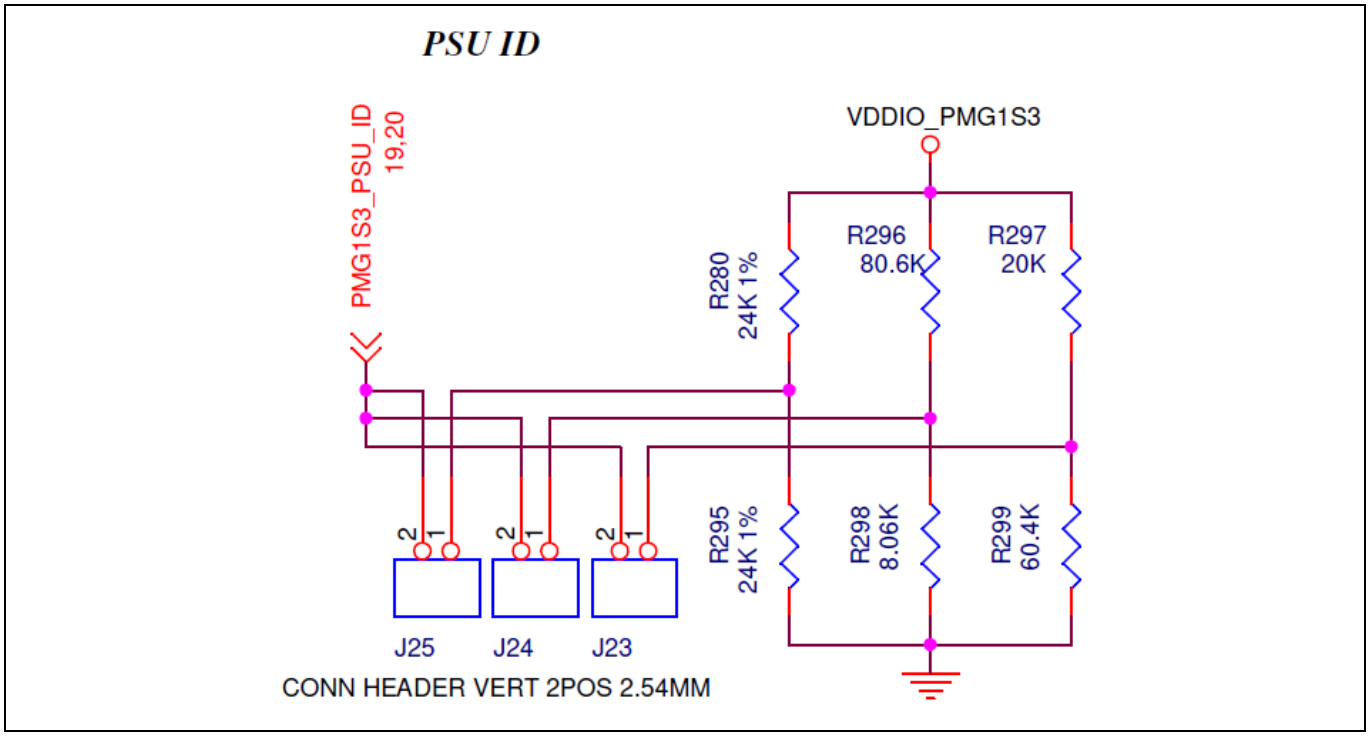

Figure 24 Combination of jumper with resistor dividers for determining power rating of adapter in PMG1-S3 USB-C Dock board

5.2.1.2 ETAG

The ETAG is a set of custom information stored in the external SPI flash memory at address '0x20000'. The ETAG space can be used for storing unique information associated with the dock such as the MAC Address, device serial number or the factory programming status of the dock. ETAG data space shown in the below figure is one time programmable through a HID command (for more details, see the ETAG update). The ETAG_VALID field is set by the firmware to indicate the validity of the ETAG_DATA space.

Figure 25 shows the layout of ETAG section reserved in the SPI flash memory.

Figure 25 Layout of ETAG section reserved in SPI flash

- ETAG_VALID: This byte is used to indicate the validity of the ETAG_DATA space. A value of '1' indicates the valid information and '0' indicates the invalid information.
- FACTORY_UPDATE_DONE: The firmware uses this byte to indicate the factory programming state of the dock. A value of '1' indicates that factory programming is completed and a value of '0' indicates that factory programming is not done.
- ETAG_DATA: 254 bytes are reserved for storing user specific ETAG information.

ETAG feature can be enabled/disabled by using the macro ETAG_SUPPORT_ENABLE. If the feature is enabled, then ETAG data space should be programmed only after the factory programming is completed. After the ETAG data space is programmed, the version check rules will apply for updating the firmware of the components in the dock.

Note: This feature is only applicable for the PMG1-S3 USB-C Dock Solution.

5.2.1.3 Extended alert events

The extended alert event features allow the dock to exchange information about the following events with the connected host:

- Sending below events to host:
  - Power button press
  - Power button release
  - Controller initiated wake
- Handling the following events received from the host:
  - Power state change
  - Power state indicator

The extended alert PD messages are supported from USBPD Specification Revision 3.1 and Version 1.1 onwards. If the port partner complies with this or higher version of the USBPD Specification, then these events are exchanged using PD messages. Otherwise, these events are propagated to the Application layer/Alternate mode layer for appropriate handling.

5.2.2 Middleware

5.2.2.1 PDStack

5.2.2.1.1 Overview

The PDStack middleware consists of:

- **USB Type-C Connection Manager**: A state machine which controls the terminations on the CC pins and monitors the CC and VBus line voltages to detect and handle connection state changes. The Type-C Connection Manager can be configured to implement various device types such as Sink, Source and Dual-Role Port.
- **USBPD Protocol Layer**: This module is concerned with sending/receiving messages through the PD transceiver
- **USBPD Policy Engine**: This layer implements the Policy Engine as defined by the USB Power Delivery specification and is responsible for initiating/responding to various PD atomic message sequences (AMS)
- **The Device/Port Policy Manager Interface** is an interface layer which allows the application to configure, initialize, monitor, and control the PDStack operation

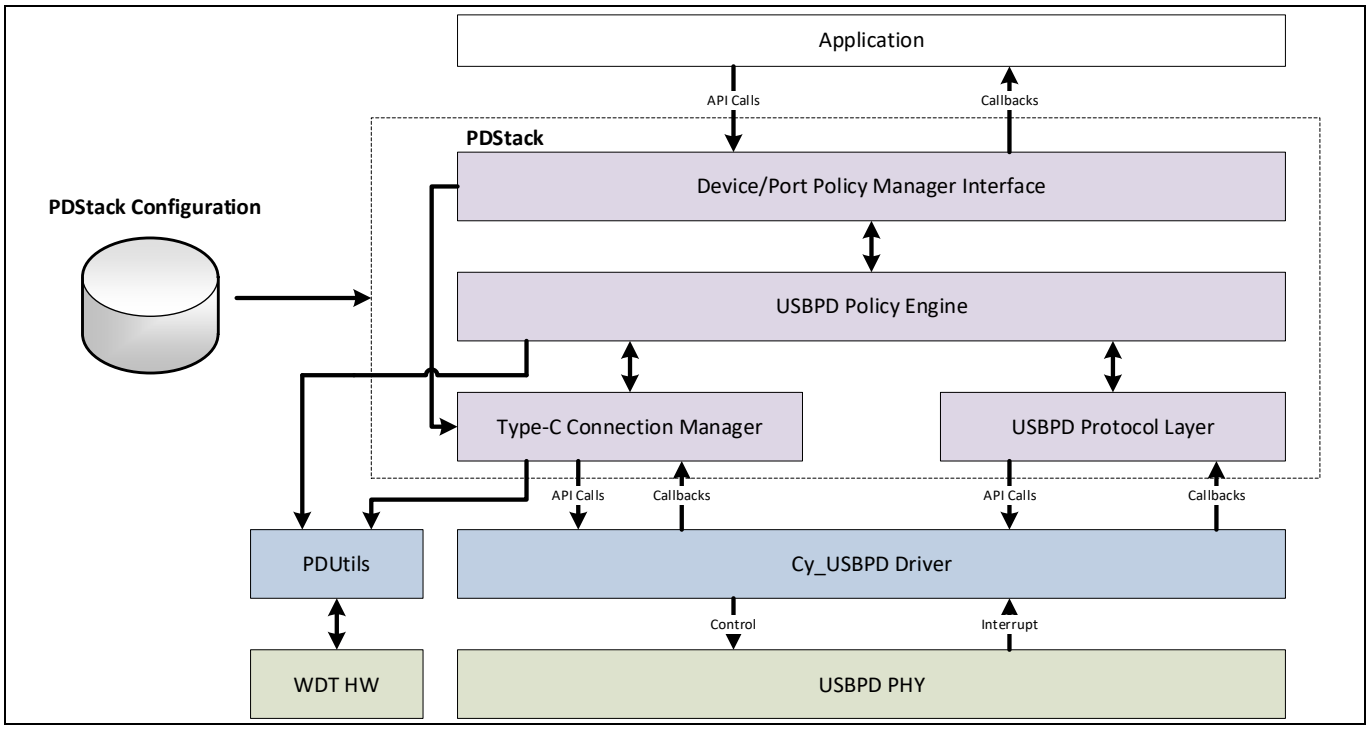

Figure 26 USBPD operation

5.2.2.1.2 PDStack variants

The PDStack middleware supports multiple libraries, with differing feature set. See the PDStack GitHub repo for the details of the variants.

The dock solutions use the “pd3_drp_epr_cfg” library variant that supports the following features:

Table 27 Features supported by pd3_drp_epr_cfg library

Features	pd3_drp_epr_cfg
Type-C sink support	Yes
Type-C source support	Yes
Type-C DRP support	Yes
Try.SRC and Try.SNK support	Yes
VCONN supply support	Yes
PD revision 3.1	Yes
PD revision 2.0	Yes
Extended power range (EPR)	Yes
Type-C cable discovery	Yes
Power role swap	Yes
Data role swap	Yes
Fast role swap	No
Configuration table support	Yes

Even though the stack is released in the library form, it provides a mechanism to configure the behavior of the stack through a set of configuration parameters. These parameters are part of the configuration space of the firmware and are generated by the EZ-PD™ Dock Configuration Tool.

5.2.2.2 USBDEV

The USBDEV middleware provides a Full speed USB 2.0 Chapter 9 specification compliant device framework. It comprises of USB device and class layers and makes use of the USBFS driver from the mtb-pdl-cat2 library. The USB device stack is developed in a layered architecture as shown in Figure 27.

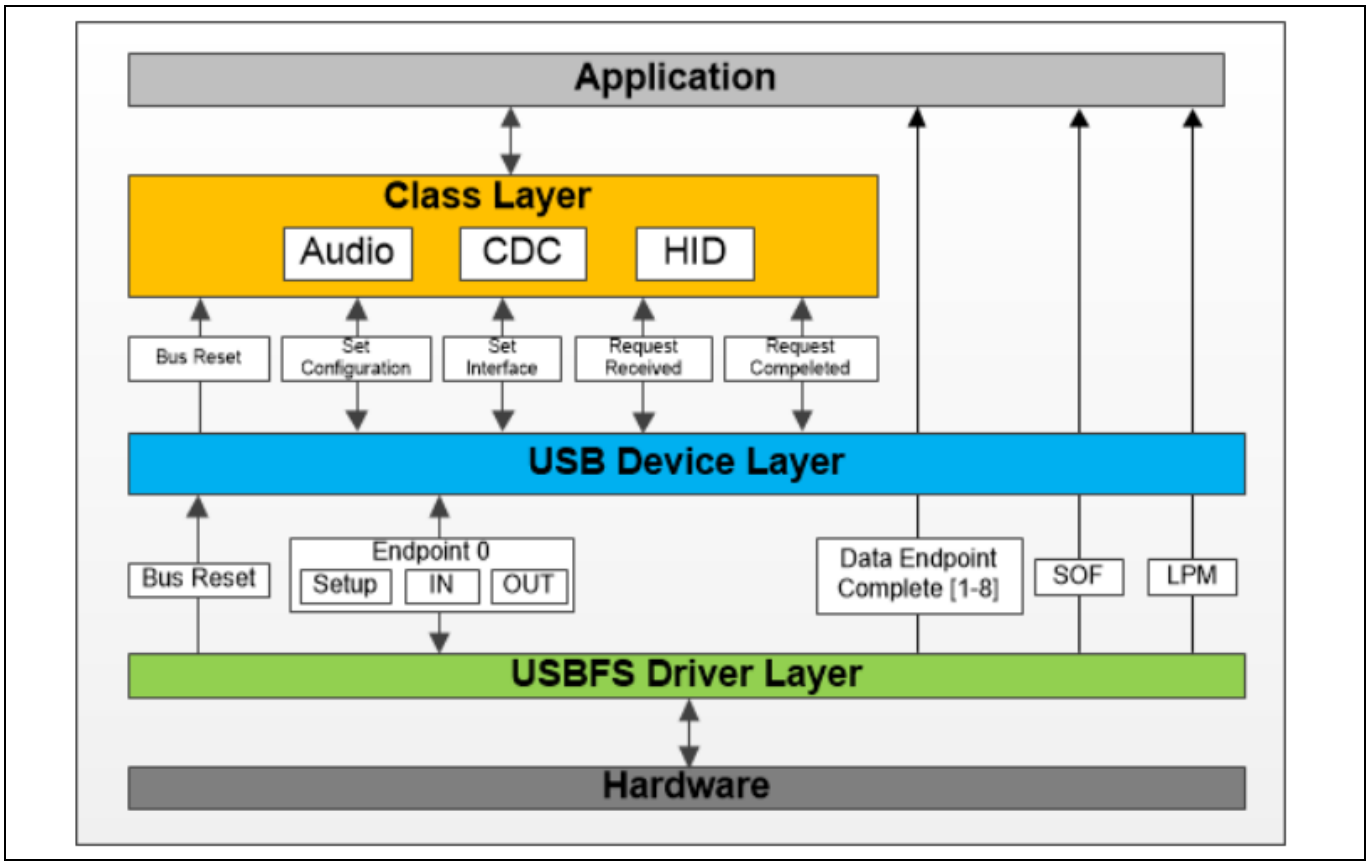

Figure 27 USB device stack

5.2.2.2.1 USB device layer

The USB device layer is responsible for handling the events passed onto it by the driver layer. It performs the task of handling USB requests that are class independent, for example, handling the Get Device Descriptor request. The device layer passes any class-specific event to the Class layer. The Device layer exposes functions to read/write from an endpoint and some service APIs to the layers on top of it to manage USB communication. In addition to that, the device layer exposes some class support functions that can be used by the class layer to register the device class to be supported and callbacks relevant to it. The device layers also expose APIs to handle vendor-specific requests.

5.2.2.2.2 Class layer

The class layer implements all the USB device class-specific functions and handles relevant requests. The current implementation of the library supports three USB device classes, namely, HID, CDC, and audio. The class layer supports user-defined callbacks for the application layer to use. The PMG1-S3 Dock SDK makes use of USBDEV middleware to support a HID class. Support for Billboard and vendor interfaces is enabled using the PmgAppCommon middleware.

HID class

The HID interface is used to send custom commands to the USB device. The HID interface is supported by the USBDEV middleware. The HID class is currently used in the USB-C dock application for setting/fetching ETAG information, retrieving Dock composite version and Dock status.

5.2.2.3 PDutils

The PDutils middleware implements the software timer and utility functions that are needed for the PMG1 solutions. The software timer module provides a WDT/SYS_TICK-based soft timer implementation. This allows different modules to create millisecond level timers without additional hardware support.

The PDutils middleware defines ranges of timer IDs for different usage, as shown in Table 28.

Table 28 Timer IDs reserved for different sections

Section name	Section	Range
Device specific	0	0x000 – 0x0FF
PD and Type-C stack (Port 0)	1	0x100 – 0x1FF
PD and Type-C stack (Port 1)	2	0x200 – 0x2FF
Base application stack (Port 0)	3	0x300 – 0x3FF
Base application stack (Port 1)	4	0x400 – 0x4FF
Alternate mode stack (Port 0)	5	0x500 – 0x5FF
Alternate mode stack (Port 1)	6	0x600 – 0x6FF
Wireless Charging	7	0x700 – 0x7FF
Solution Layer	8	0x800 – 0x8FF
Reserved	9 – 15	–
User space	16 - 255	–

The actual timer ID allocation and management within the available range is done by the respective middleware/modules. The device-specific timer IDs are defined and managed by the PDutils asset.

5.2.2.4 Host processor interface (HPI)

The HPI protocol is used for communication with a CCGx PD controller. This protocol allows HPI master to query the device and PD status, modify configuration and update the firmware of HPI slave devices. HPI middleware v1.0 supports the HPI Master feature where PMG1-S3 is the master and CCGx is the HPI slave device.

The HPI protocol is implemented over the I2C protocol and the HPI slave devices make use of an interrupt line to notify the HPI Master about events, responses, and asynchronous messages.

Some of the high-level features supported by the HPI protocol are:

- Firmware version identification
- Firmware update capability
- Reporting of Type-C and USB PD connection status
- Interrupt-based event reporting when connection status changes
- Control of USB PD power profiles
- Send and received USB PD vendor defined messages (VDMs)
- Initiate USB PD control messages
- Initiate alternate mode entry and exit
- Provide interface to control hardware resources, such as Type-C data mux

5.2.2.5 PDAltmode

The PDAltMode middleware implements the state machines defined in:

- USB Power Delivery specification
- VESA DisplayPort Alt Mode on USB Type-C standard
- Thunderbolt interconnect specification
- Goshen Ridge Thunderbolt/USB4 controller

The middleware provides a set of alternate mode APIs through which the application can initialize, monitor, and configure different VDM Alt Modes:

- Display port
- TBT
- USB4

The PDAltMode middleware operates on top of the USB PD included in the MTB PDL CAT2(mtb-pdl-cat2) peripheral driver library (PDL), PDUtils, and PDStack middleware.

5.2.2.6 PmgAppCommon

The PmgAppCommon middleware library provides the following functionalities that are needed for a USB-C and Power Delivery application.

5.2.2.6.1 USB

The USB middleware does not provide support for vendor and Billboard classes. Therefore, the support for these classes is enabled on top of the USB middleware and PDL in the PmgAppCommon middleware.

The Billboard class is needed in applications that require alternate modes to be supported. In the current Dock solutions, the Billboard device does a silent enumeration if the alternate mode is successfully entered. Otherwise, the billboard device enumerates after tAMETimeout and displays a pop-up that indicates the capabilities of the device to the user.

The vendor class is used mainly for updating the firmware of DMC and other connected components in the Dock. It can be also used for handling custom vendor commands sent from the USB host.

5.2.2.6.2 Buck-boost

The buck-boost driver implements an abstraction layer for communicating with the external buck-boost controllers. The PmgAppCommon middleware supports drivers forMP4247 from MPS and RT6190 from Richtek. Only one controller can be enabled at a time. To enable the desired controller driver, the respective feature macro should be enabled, that is, CY_APP_BUCKBOOST_MP4247_ENABLE for MP4247 or CY_APP_BUCKBOOST_RT6190_ENABLE for RT6190.

The buck-boost controller drivers communicate with an external buck boost controller using the standard I2C communication protocol to enable and configure the controllers to output desired power levels. This is achieved by writing to a set of registers provided by the external buck boost controller.

See the EZ-PD™ PMG1-S3 Dock user guide for more information.

5.2.2.6.3 Debug Module

The debug module provides UART-based event logging and logging of events of interest to the internal flash of PMG1-S3. The events that are logged are broadly classified as below:

- Common errors
- DMC errors
- PD port errors (Port 0 and Port 1)
- Logs for tracking firmware events

The details about the above events are described below:

1. **Common errors**

Table 29 Common errors

Error	Description
WDT	Watchdog reset triggered
Hard Fault	Triggered a hard fault
Power cycle/reset	Power cycle or reset event detected
VDDD brownout fault	VDDD brown out fault detected. (Not applicable for PMG1-S3)

2. **DMC errors**

Table 30 DMC errors

Error	Description
FW update failure	Firmware update failed
SCB failure	SCB related failure occurred
HPI event	HPI event received

3. **PD port errors (Port 0 and Port 1)**

Table 31 PD port errors

Error	Description
VBUS overvoltage (VBUS OV)	Type-C VBUS overvoltage fault
VBUS undervoltage (VBUS UV)	Type-C VBUS undervoltage fault
VBUS short-circuit (VBUS SC)	Type-C VBUS short-circuit fault
VBUS ove current (VBUS OC)	Type-C VBUS overcurrent fault
VBUS reverse current (VBUS RC)	Type-C VBUS reverse-current fault
VCONN overcurrent (VCONN OC)	VCONN overcurrent fault
VBUS_IN overvoltage (VBUS_IN OV)	VBUS_IN overvoltage fault (not applicable for PMG1-S3)
VBUS_IN undervoltage (VBUS_IN UV)	VBUS_IN undervoltage fault (not applicable for PMG1-S3)
System overtemperature (System OT)	System overtemperature fault (not applicable for PMG1-S3)
Cyclic redundancy check (CRC) error	CRC error in the PD message
CC overvoltage (CC OV)	Overvoltage fault on CC lines (not applicable for PMG1-S3)
CC short-circuit (CC SC)	Short-circuit fault on CC lines (not applicable for PMG1-S3)
SBU overvoltage (SBU OV)	Overvoltage fault on SBU line (not applicable for PMG1-S3)
Type-C attach	Type C connection event
Type-C detach	Type C disconnect event
Hard reset (transmit)	PD Hard reset transmitted by the port
Hard reset (receive)	PD Hard reset received by the port
Port enable	Type C port enabled
Port disable	Type C port disabled

4. **Logs for tracking following firmware events**

Table 32 Firmware events

Error	Description
Phase 2 update start	Indicates the start of phase 2 of firmware update
Phase 2 authentication success	Indicates success of phase 2 authentication
Phase 2 image write success	Indicates success of write operation in phase 2
Phase 2 image write fail	Indicates failure of write operation in phase 2
Pending updates	Indicates firmware update pending
Phase 2 image update start	Indicates image update start in phase 2 of firmware update
Phase 2 factory backup started	Indicates start of factory backup in phase 2 of firmware update
Phase 2 dock reset	Indicates dock reset event in phase 2 of firmware update
Phase 2 factory backup not done	Indicates factory backup incomplete in phase 2 of firmware update
Phase 2 factory update status failed	Indicates factory backup failure in phase 2 of firmware update
DMC state	Indicates the current DMC state
HPI Master event received	Indicates the reception of an HPI master event
HPI Master queue push error	Indicates HPI master queue push error
Trace info	Indicates any trace information for debug (for example, line number)
PA size	Power adapter size
Current running image	Current running image ID to distinguish primary and backup image

Logging through UART

This feature enables you to transmit log messages over the UART interface. The following is the structure of the log message sent out over the UART:

Table 33 UART log message structure

Standard/custom	Reserved	Size of variable field in bytes	Opcode	Variable in bytes as per size field
1 bit	4 bits	3 bits	2 bytes	0-7 bytes

Details regarding each of the fields in the UART log message are given below:

- Standard/custom- Standard refers to the UART logs coming out the different modules of the Dock solution. The custom refers to the additional prints that can be defined by the customers.
- Size of variable field in bytes – This field is used to add any additional debugging information needed for debugging the error.
- Opcode – 2-byte opcode used to identify the error.
- Variable ‘n’ bytes as per size field – The additional debugging data on top of the error opcode.

A delimiter string “\r\n” is sent after each log message is transmitted over UART. This delimiter is used to mark the end of each debug log sent over UART.

After initialization of the UART interface, a “UART INIT” string is sent to mark the start of UART logging.

The UART debug module also makes use of different log levels to determine the priority of a log message. The following log levels are used for the implementation of UART debug logs:

- a. LOGLEVEL_CRITICAL (highest priority)
- b. LOGLEVEL_ERROR
- c. LOGLEVEL_WARNING
- d. LOGLEVEL_INFO (lowest priority)

The compile-time option **CY_APP_DEBUG_LEVEL** along with the log levels determine if a UART log needs to be transmitted or not.

Storing into flash

This feature supports logging of static as well as dynamic information into internal flash. The layout of the internal flash for logging this information is shown below:

Table 34 Layout of internal flash for logging

Flash Row 1 (for static information) Row number: 0x3F5	Header (16 bytes)	Common Faults (32 bytes)	DMC (64 bytes)	Port 0 (72 bytes)	Port 1 (72 bytes)
Flash row 2 (For dynamic information) Row number: 0x3F6	Failure details of last 85 errors (255 bytes)				Check sum (1 byte)

Note: *Contact Infineon for assistance in decoding the debug logs.*

5.2.2.6.4 LED control module

This module can be used to control a user LED with a user selected I/O.

Table 35 Modes supported by LED control module

Mode	Description
OFF	LED remains off
ON	LED remains on
Blink	LED blinks at a fixed rate
Breathing	Slow ramp up and down of LED brightness to give a visual snoozing effect on LED

5.2.2.6.5 Dock firmware update

The Dock firmware update module is responsible for updating the firmware on PMG1-S3 and connected components such as Goshen Ridge, Foxville, and CCGx present on the dock board. The DMC supports both signed and unsigned firmware updates and provides a mechanism to switch from unsigned to signed without having to recompile the firmware. Once switched to signed, unsigned firmware updates are not allowed and are failed. The firmware of DMC and its components is updated over two phases:

Phase 1 – In this phase, a composite image consisting of firmware images for PMG1-S3, and its components are downloaded from a USB Host to the external SPI Flash connected to PMG1-S3 over a USB Vendor interface using the EZ-PD™ Firmware Update Utility. If the firmware is operating in signed mode, it verifies the integrity of the image being downloaded using the SHA-256 and RSA-2048 Crypto algorithms.

Phase 2 – This phase is triggered by the EZ-PD™ Firmware Update Utility when it sends a special command to PMG1-S3 to initiate the firmware update of PMG1-S3 and its connected components. PMG1-S3 queries the connected components to determine if a firmware update is needed. The image(s) in the SPI flash are verified for integrity prior to updating the firmware on PMG1-S3 and other components. PMG1-S3 may also perform factory updates from the primary composite image if requested and resets itself at the end of this phase.

Refer to EZ-PD™ PMG1-S3 Dock user guide for more information.

5.2.2.6.6 Fault protection

The Fault protection provides handlers for fault situations on the VBus or VCONN voltage rails and trips the supply to avoid damage to any of the system components. The module monitors the voltages and compares with threshold values.

The fault handler is built over the USB PD driver of the mtb-pdl-cat2 library. The following fault protections are supported in the Dock solution:

- VBUS, overvoltage protection
- VBUS, undervoltage protection
- VBUS, short-circuit protection
- VBUS, overcurrent protection
- VCONN, overcurrent protection

5.2.2.6.7 I2C master

The I2C master is implemented over the SCB I2C driver of the mtb-pdl-cat2 library and provides a set of APIs for transferring data to/from I2C slave device registers.

5.2.2.6.8 PDO handling

The PDO handler provides APIs to evaluate the source capabilities advertised by the port partner. And identify the optimal power contract to be entered. It also evaluates a PD Request Data Object and determine whether to accept or reject the request.

5.2.2.6.9 Consumer path

The consumer path handler provides APIs and data structures to control the power consumer path and enables the fault detection.

These APIs are registered as application callbacks and are invoked by the PDStack middleware library.

The key features are:

- Enable/disable consumer path
- Set fault threshold voltage

5.2.2.6.10 Provider path

The provider path handler provides APIs and data structures to control the power provider path and enables the fault detection.

These APIs are registered as application callbacks and are invoked by the PDStack middleware library.

The key features are:

- Set fault threshold voltage
- Enable/disable the provider path
- Set VBUS voltage and current

5.2.2.6.11 Smart power

The Dock solution uses the smart power module to perform power throttling on the USB-C ports depending on the available power in the dock. PMG1-S3 periodically monitors the power consumed by the dock and adjusts the power on the upstream port depending on the available power. The smart power is released in the library form and provides a set of parameters that can configure the behavior using the EZ-PD™ Dock Configuration Utility. Figure 28 shows the smart power configurable parameters.

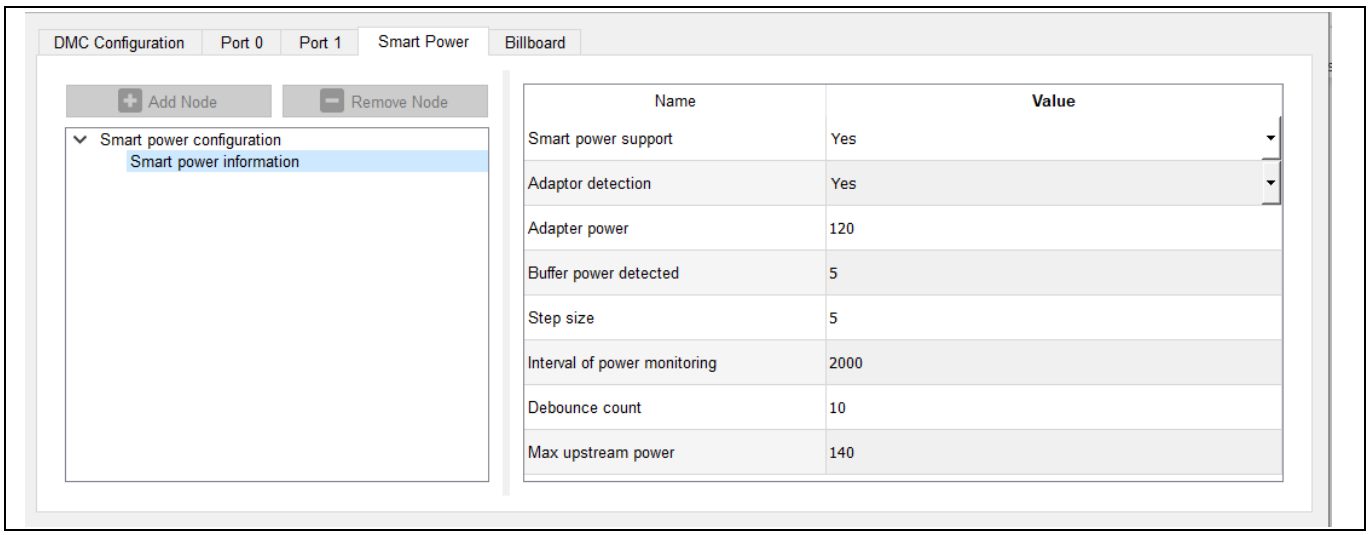

Figure 28 Smart power configurable parameters

5.2.2.6.12 Swap handling

The swap handler provides APIs and data structures for evaluating the swap requests.

These APIs are registered as application callbacks and are invoked by the PDStack middleware library.

The supported swap requests are:

- Data role swap
- Power role swap
- VCONN swap

5.2.2.6.13 VDM

The vendor defined message (VDM) handler provides APIs and data structures to evaluate the VDM command and send response.

These APIs are registered as application callbacks and are invoked by the PdStack middleware library.

The supported features are:

- Evaluate VDM command
- Initialize and send VDM response

5.2.2.6.14 System

The system module provides:

- A set of APIs to set/get the firmware mode, metadata version, firmware version information
- APIs and data structures to validate and load the firmware image from flash
- APIs to perform reads and writes from/to the device flash memory
- APIs to monitor the CPU resource (execution time and stack) usage

5.2.3 mtb-pdl-cat2 PDL

The mtb-pdl-cat2 PDL integrates device header files, startup code, and peripheral drivers into a single package. The drivers abstract the hardware functions into a set of easy-to-use APIs. These are fully documented in the PDL API reference manual.

5.3 SDK usage model

Follow these steps to use the SDK components:

1. Create the project using any method mentioned in the Project creation.
2. If needed, edit the *design.modus* file based on the requirements. This can be done by using the Device Configurator tool, which is available as part of the Quick Panel inside Eclipse IDE for ModusToolbox™ software (See the section 6.1.1 and section 6.2.1).
3. Use the EZ-PD™ Dock Configuration Utility to build the configuration table and copy the generated C source file into the Eclipse IDE for ModusToolbox™ software. The configuration table can also be updated by editing the config.c file in the source editor.
4. Build the project using any method mentioned in the Compiling the reference project. The firmware binary will be generated in hex format suitable for SWD programming.

5. For evaluation and testing, load the firmware binary onto the target hardware using any method mentioned in the Programming PMG1-S3 on the dock.

5.4 Firmware versioning

Each project has a firmware version (version) and an application version number.

The base firmware version must consist of a major number, minor number, and patch number in addition to an automatically updated build number.

The base firmware version applies to the whole stack and is common for all applications and projects using the stack. The version information can be found in the *src/pmg1_version.h* header file.

The application version shall be modified for individual customers based on requirements. This shall have a major version, minor version, circular version number, and a signature string. This version information can be updated as required by users and is located in the *src/app_version.h* header file.

5.5 Bootstrap

The bootstrap is the first image that runs when the PMG1-S3 is powered up. The Firmware architecture shows the layout, bootstrap, and firmware images in the internal flash of PMG1-S3. The bootstrap performs integrity checks of the available images and transfers control to the requested image if it is valid. If the images are not valid, then the control remains in the bootstrap. This is indicated by driving P2.7 of PMG1-S3 LOW.

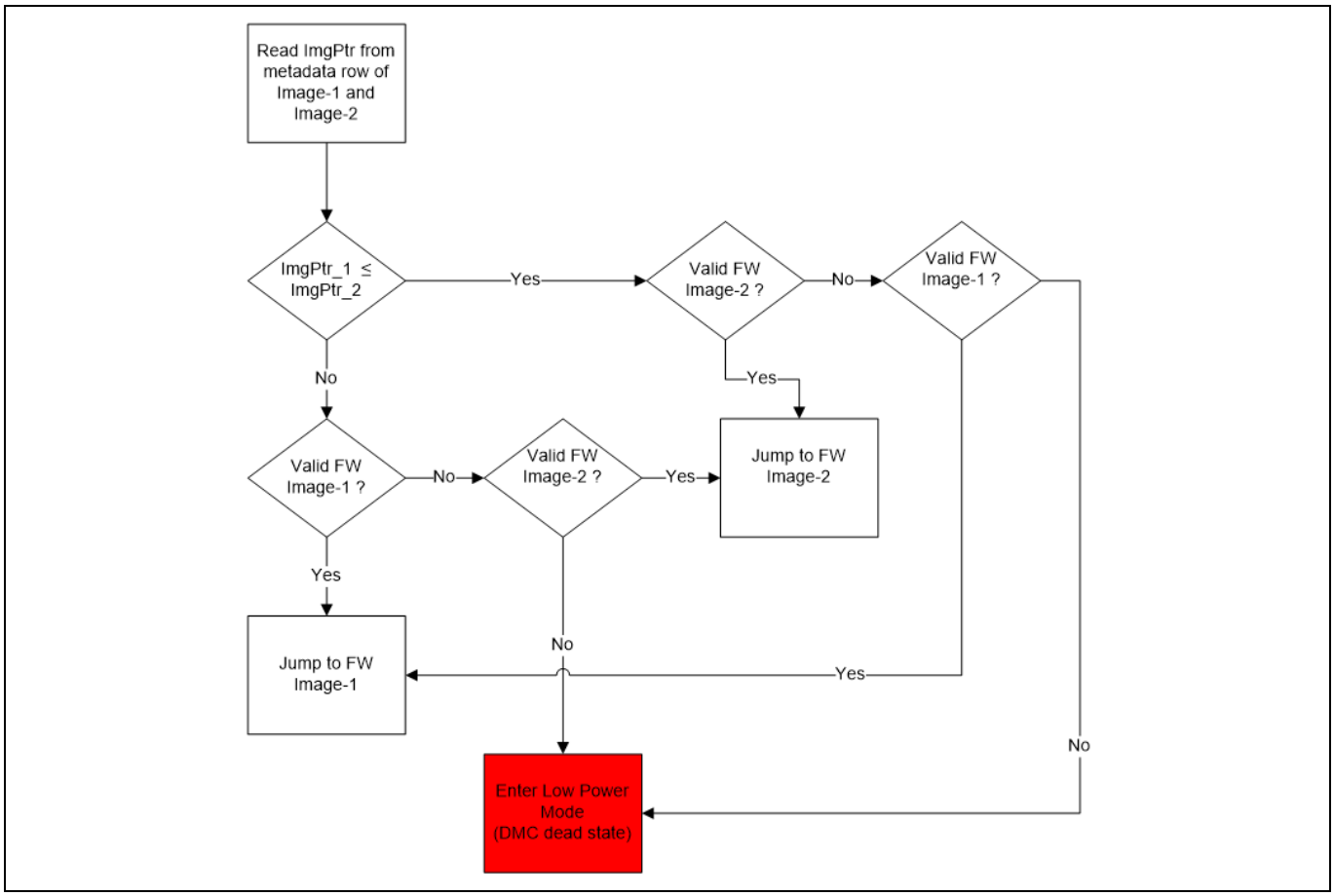

Figure 29 Firmware flow of bootstrap

5.6 Firmware operation

The Dock firmware is implemented in the form of a set of state machines and tasks that need to be performed periodically.

The code flow for the application is implemented in the *main.c* file. As can be seen from the `main()` function, the implementation is a simple round-robin loop, which services each of the tasks that the application has to perform.

The PD management, HPI command handling, Flash log handling, HID handling, and VDM handling are encapsulated in the task handlers in the PMG1-S3 firmware stack. See the Firmware API Guide for more details of these functions and handlers.

Figure 30 shows the firmware initialization and operation sequence.

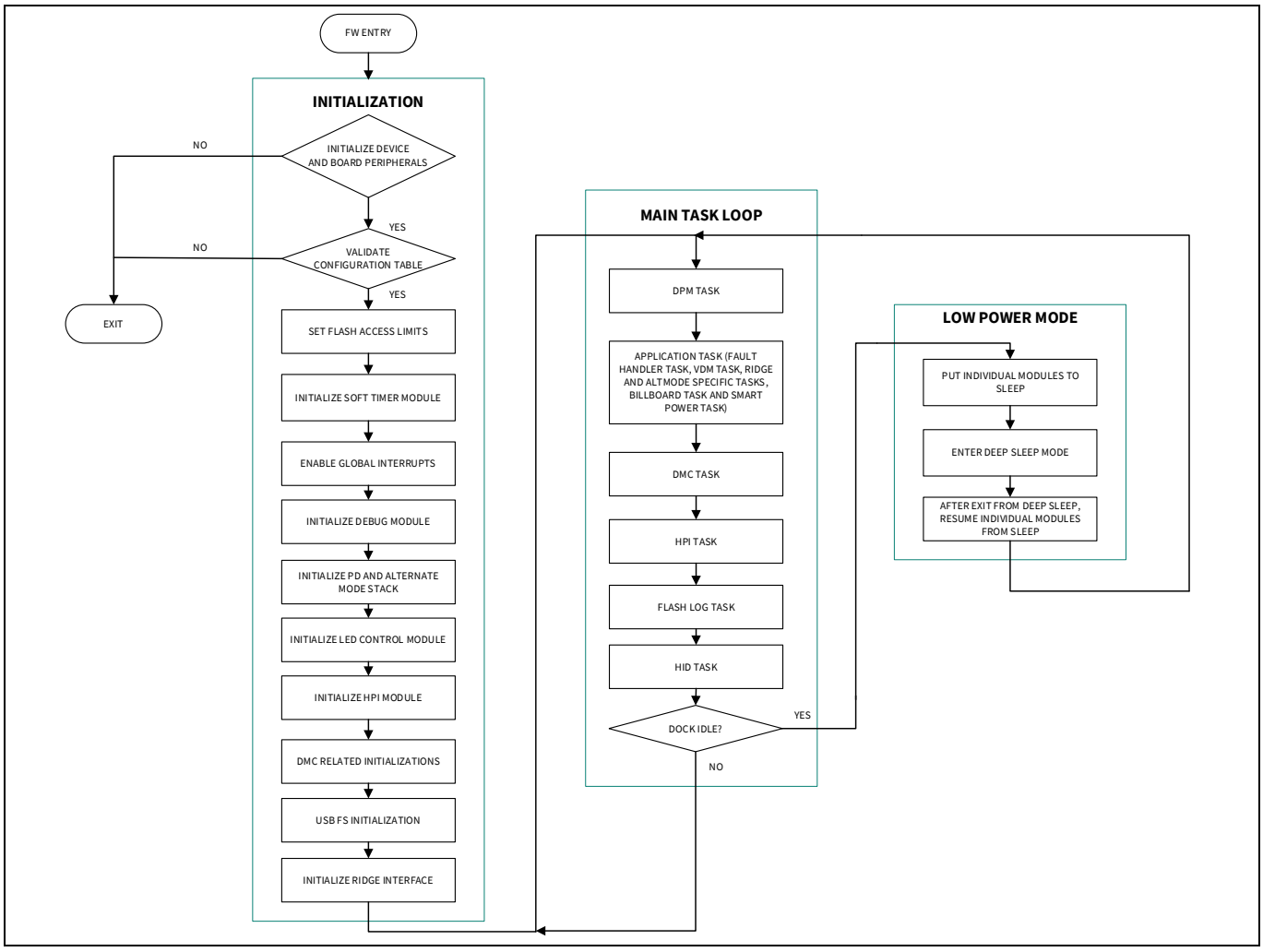

Figure 30 Firmware initialization and operation sequence

Firmware architecture

The required initializations are performed as part of the `main()` function. The main steps involved in the initialization stage are:

1. **Initialize the device and board peripherals**: Initializes the system clocks, peripherals, and pins of PMG1-S3 after the firmware enters the main function.
2. **Validate the configuration table offsets**: Reads the configuration table and validates the offsets of the configuration table for both ports.
3. **Set flash access limits**: Sets the legal flash access range for primary and backup images.
4. **Initialize the soft timer module**: In this step, an interrupt handler is registered for the watchdog timer. This timer is used to implement soft timers required by the firmware blocks. After registering the interrupt handler for the watchdog timer, the soft timer module is initialized.
5. **Initialize debug module**: Initializes the debug module to log events or failures that would occur during firmware execution.
6. **Initialize PD and alternate mode stack**: Configures USBPD interrupts and invokes appropriate APIs to initialize PDStack and Alternate mode middleware. This is needed to enable power delivery, fault protections, and alternate mode operations for the Dock solution.
7. **Initialize the LED control module**: The LED control module is used to control LEDs connected to the GPIOs of PMG1-S3. Details such as the port and pin on which the LED control needs to be done, timer ID used by the module, LED connection orientation, and blink period are passed as parameters while initializing this module.
8. **Initialize the HPI module**: HPI module used to communicate with other CCGx devices is initialized in this step. This step is only needed for the EZ-PD™ PMG1-S3 USB-C Dock Solution.
9. **DMC-related initializations**: Initializes the DMC module for firmware update operations. In this step, various callback functions are registered to the DMC module for its operation.
10. **USBFS initialization**: Initializes the USB module with the required context structures, configuration structures and callback functions for its internal operation and enables it.
11. **Initialize Ridge interface**: Initializes the Ridge slave interface module and configures it to use the specified SCB block. This step is needed only for EZ-PD™ PMG1-S3 Griffin Creek Dock solution.

After the required initializations, the main task loop is run infinitely. The main task loop runs the following other tasks:

- **DPM task**: Runs the Type-C manager and PD policy manager tasks for a port.
- **Application task**: Internally handles the following application tasks:
  - **Fault handler task**: Handles any application events associated with fault handling logic.
  - **VDM task**: The application checks if VDM processing is allowed. If the VDM processing is allowed, it passes control to the PDAltmode middleware to handle VDMs.
  - **Ridge and altmode specific tasks**: These tasks handle pending tasks related to ridge and exchange capabilities for the Dock solution.
  - **Billboard task**: This task monitors the billboard state and enables/disables the USB interface depending on the inputs received from the PDAltmode middleware.
  - **Smart power task**: This task monitors the power consumed by the dock and updates the power advertised to the upstream port.
- **DMC task**: Handles the firmware updates for the connected devices and DMC.
- **HPI task**: Handles the HPI events received from the HPI slave devices. HPI port events from the slave are handled in the interrupt context and any associated data is queued to be handled by this function.
- **Flash log task**: This task is used for internal flash logging. It checks for pending flash writes, defer the writes, and finally write the logs to the flash. It also handles the logging of backup logs into internal flash.

- **HID task**: This task is relevant only for the EZ-PD™ PMG1-S3 USB-C Dock Solution. It runs a state machine for custom report exchange between the host and the dock. It also handles the transmission of system control reports to the host.

The application tries to keep the PMG1-S3 MCU device in Deep Sleep, where all clocks are disabled and only limited hardware blocks are enabled, for most of its working time. Wake-up interrupts are configured to detect any changes that happen while the device is in Deep Sleep and wake it up for further processing.

5.6.1 Fault handling

PMG1-S3 supports different forms of fault detection and handling capabilities. The following table summarizes the various kinds of fault detection and handling supported in various applications in the SDK.

Table 36 Fault detection and handling

Type of fault	Comments
VBUS overvoltage	Hard reset and recovery will be attempted for a configurable number of retries. The firmware suspends the port and waits for physical disconnection after all retries have elapsed.
VBUS overcurrent	Hard reset and recovery will be attempted for a configurable number of retries. The firmware suspends the port and waits for physical disconnection after all retries have elapsed.
VBUS short-circuit	Hard Reset and recovery will be attempted till a configurable number of retries. The firmware suspends the port and waits for physical disconnection after all retries have elapsed.
VCONN overcurrent	The firmware exits any alternate modes that require VCONN to be present. VCONN will be reenabled after a delay, if retries are enabled.
VBUS undervoltage	Hard reset and recovery will be attempted for a configurable number of retries. The firmware suspends the port and waits for physical disconnection after all retries have elapsed.

6 Customizing the firmware application

6.1 EZ-PD™ PMG1-S3 USB-C Dock solution

6.1.1 Device configurator

Most aspects of the hardware design around the PMG1-S3 device are captured in the *design.modus* file associated with the ModusToolbox™ project. The *design.modus* file can be opened from Eclipse IDE for ModusToolbox™ software by clicking **Device Configurator** under **Quick Panel** (see Figure 31).

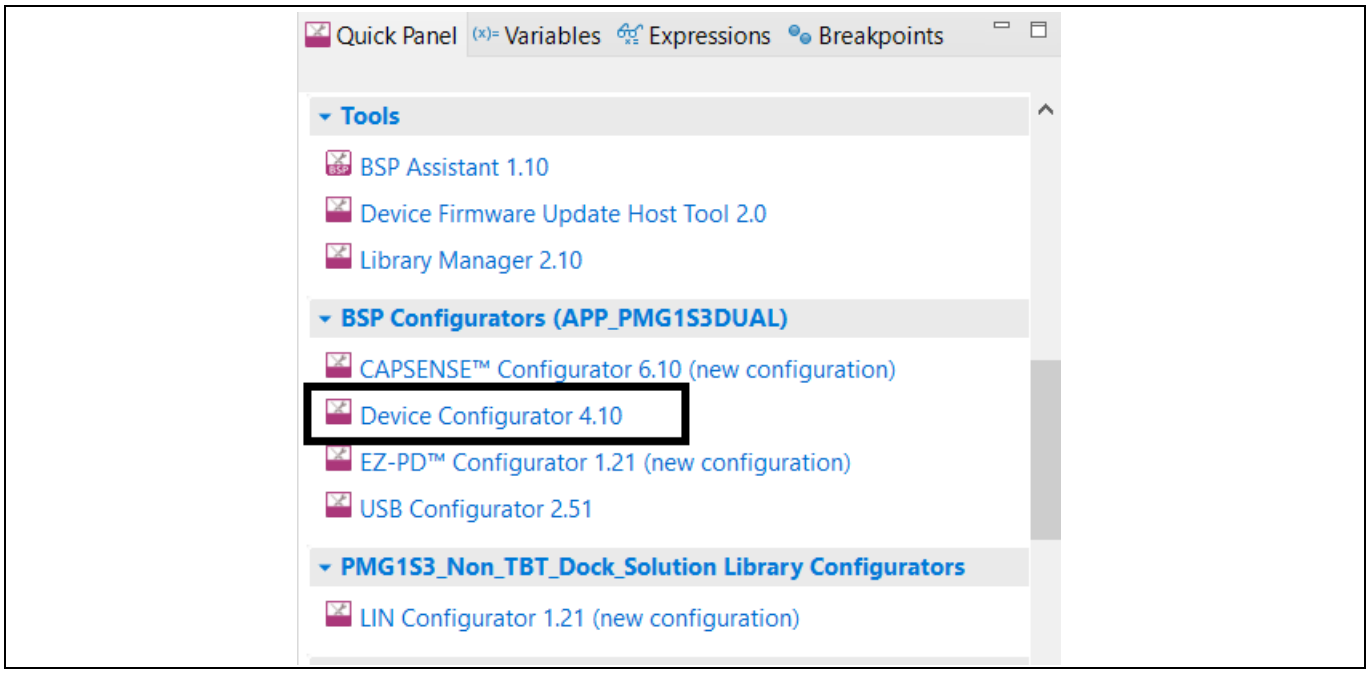

Figure 31 Launching Device Configurator from Eclipse IDE for ModusToolbox™ software

The *design.modus* file is used to configure the internal resources of the PMG1-S3 device that needs to be used in the design. This includes all the internal clocks used by the design, the various serial interfaces, and all the GPIO pins used to communicate with external elements.

Table 37 lists the various resources of PMG1-S3 used in the EZ-PD™ PMG1-S3 USB-C Dock solution. The selection of some of these resources is fixed due to the capabilities of the PMG1-S3 device. The table also points out the changes allowed in the default design.

Table 37 Resources of PMG1-S3 used in EZ-PD™ PMG1-S3 USB-C Dock solution

Peripheral	Pin	Name	Description	Changes allowed
GPIO	P1.0	HX3_VBUS_US	GPIO used by the HX3 mux to detect the US VBUS status.	It can be changed to a different pin
SWD	P1.1	CYBSP_SWCLK	SWD clock for clocking the data	No changes are allowed
SWD	P1.2	CYBSP_SWDIO	SWDIO pin for data transfer	No changes are allowed

Peripheral	Pin	Name	Description	Changes allowed
GPIO	P2.0	LAN_WAKE_BTN	The button used for sending extended alert event “Controller Initiated Wake”	It can be changed to make use of a different input pin
GPIO	P2.1	PFET_SNK_CTRL_P0	Used for SINK consumer path FET control	It can be changed to a different pin
SCB5 (UART)	P2.2	CYBSP_DEBUG_UART_TX	Used for transmitting UART debug logs	It can be changed to a different SCB
GPIO	P2.6	CUR_SENSE_SYS	Used as an analog pin to measure system current	It can be changed to a different pin
GPIO	P3.0	LED_CTRL	This pin will be connected to an onboard LED. This pin will be used by the LED control module to control the LED.	It can be changed to a different output pin to which an LED is connected
GPIO	P3.2	HX3_RESETN	GPIO used for HX3 USB hub reset pin	It can be changed to a different pin
GPIO	P3.3	PSU_ID	This pin will be used for detecting the adaptor power	It can be changed to a different pin
GPIO	P3.4	CUR_SENSE_US	Used as an analog pin to measure US port current	It can be changed to a different pin
SCB4 (I2C)	P3.5	I2C_PWR_MUX_SCL	Used as SCB4 I2C bus SCL pin	It can be changed to a different SCB
SCB4 (I2C)	P3.6	I2C_PWR_MUX_SDA	Used as SCB4 I2C bus SDA pin	It can be changed to a different SCB
SCB0 (I2C)	P4.0	I2C_HPIM_SCL	Used as SCB0 I2C bus SCL pin	It can be changed to a different SCB
SCB0 (I2C)	P4.1	I2C_HPIM_SDA	Used as SCB0 I2C bus SDA pin	It can be changed to a different SCB
GPIO	P5.2	REG_EN	Used for US port regulator enable pin	It can be changed to make use of a different input pin
GPIO	P5.3	VMM_RESETN	User for display MUX reset pin	It can be changed to make use of a different input pin
GPIO	P7.2	POWER_BTN	Used to send “Button Press” and “Button Release” extended alert events	It can be changed to make use of a different input pin
SCB7 (SPI)	P7.3	FLASH_SPI_MISO	MISO line for the SPI interface used to communicate with external flash	It can be changed to a different SCB

Peripheral	Pin	Name	Description	Changes allowed
SCB7 (SPI)	P7.4	FLASH_SPI_CLK	Clock line for the SPI interface used to communicate with external flash	It can be changed to a different SCB.
SCB7 (SPI)	P7.5	FLASH_SPI_MOSI	MOSI line for the SPI interface used to communicate with external flash	It can be changed to a different SCB
SCB7 (SPI)	P7.6	FLASH_SPI_CS	Chip select line for the SPI interface used to communicate with external flash	It can be changed to a different SCB
USBFS	P8.0	USBDP	DP line for USB FS interface	No changes are allowed
USBFS	P8.1	USBDM	DM line for USB FS interface	No changes are allowed
PD Port 0	–	PD_PORT0	USB Power Delivery Port 0	No changes are allowed
PD Port 1	–	PD_PORT1	USB Power Delivery Port 1	No changes are allowed

Note:

1. *Any modifications done to the **Peripherals** and **Pins** tab of the Device Configurator should be in accordance with the **Changes allowed** field in the table given above.*
2. *If the EZ-PD™ PMG1-S3 USB-C Dock Board is used as hardware, it is recommended not to change any resources mentioned in the table given above. For more details on modifying these tabs, see the Device Configurator user guide.*

6.1.2 Compile-time options

The PMG1-S3 Dock management controller application supports a set of features that can be enabled/disabled using compile-time options. These compile-time options are set in the Makefile. These options are summarized in Table 38.

Table 38 Compile time options for EZ-PD™ PMG1-S3 USB-C Dock solution

Option	Description	Values	Change allowed
CY_APP_LED_CONTROL_ENABLE	Enable/disable LED control module.	1: Enable the LED control module 0: Disable the LED control module	Allowed
CY_USE_CONFIG_TABLE	Enable/disable configuration table support.	1: Enable the configuration table 0: Disable the configuration table.	Not allowed

Option	Description	Values	Change allowed
			Note: *Dock solution makes use of the configuration table.*
CY_CONFIG_TABLE_TYPE	Configuration table device type	'4' for Dock configuration table	Not allowed ***Note:*** *The Config table type used by this version of the SDK is '4'.*
CY_APP_ROLE_PREFERENCE_ENABLE	Enable/disable preference for attempting to switch to the port's preferred Data and Power roles.	1: Enable the preference for attempting to switch to the port's preferred data and power roles. 0: Disable the preference for attempting to switch to the port's preferred data and power roles.	Allowed
CY_APP_POWER_ROLE_PREFERENCE_ENABLE	Enable/Disable preference for port's power role.	1: Enable the port power role preference. 0: Disable the port power role preference.	Allowed
CY_APP_PD_PDO_SEL_ALGO	PDO Selection algorithm **Note:** Applicable in sink role only.	0: Pick the Source PDO, which delivers the maximum amount of power. 1: Pick the Fixed Source PDO, which delivers the maximum amount of power. 2: Pick the Fixed Source PDO, which delivers the maximum current. 3: Pick the Fixed Source PDO, which delivers power at maximum voltage.	Allowed
CY_APP_UART_DEBUG_ENABLE	Enable/Disable option for UART logs	1: Enable the support for UART logs 0: Disable the support for UART logs	Allowed.

Option	Description	Values	Change allowed
CY_APP_RESET_ON_ERROR_ENABLE	Enable/Disable software reset on error.	1: Enable the software reset on error. 0: Disable the software reset on error.	Allowed.
CY_APP_WATCHDOG_HARDWARE_RESET_ENABLE	Enable/Disable option for watchdog reset.	1: Enable the watchdog reset. 0: Disable the watchdog reset.	Allowed.
CY_APP_FLASH_LOG_ROW_NUM	Flash address row number where the logs are stored.	0x3F5u	Allowed. ***Note:*** *Make sure that the newly chosen row is from the reserved section of flash.*
CY_APP_FLASH_LOG_BACKUP_ROW_NUM	Flash address row number where a redundant copy of the logs are stored.	0x3F7u	Allowed ***Note:*** *Make sure that the newly chosen row is from the reserved section of flash.*
CY_APP_BOOT_LOADER_LAST_ROW	Last flash row number of the bootloader image space	0x0Fu	Not allowed
CY_APP_IMG1_LAST_FLASH_ROW_NUM	Last flash row number of FW1 Image	0x010Fu	Not allowed
CY_APP_IMG2_LAST_FLASH_ROW_NUM	Last flash row number of FW2 Image	0x3EFu	Not allowed
CY_APP_PD_USB4_SUPPORT_ENABLE	Enable/Disable USB4 Support feature at the Application level.	1: Enable the USB4 support feature at the application level. 0: Disable the USB4 support feature at the application level.	Not allowed
CY_APP_DMC_ENABLE	Enable/Disable DMC support.	1: Enable DMC support 0: Disable DMC support	Not allowed
BATTERY_CHARGING_ENABLE	Enable/Disable legacy battery charging support.	1: Enable legacy charging support 0: Disable legacy charging support	Not allowed ***Note:*** *This feature is not supported by the*

Option	Description	Values	Change allowed
			PMG1-S3 dock solution.
VBUS_C_DISCHG_DS	Set VBus discharge drive strength.	Value to be set from 1 to 16	Not allowed. ***Note:*** *It is not recommended to change this option.*
TBT_DFP_SUPP	Enable/Disable option for TBT alternate mode in DFP port.	1: Enable the TBT mode in DFP 0: Disable the TBT mode in DFP	Not allowed ***Note:*** *USB-C Dock does not support the TBT alternate mode.*
TBT_UFP_SUPP	Enable/Disable option for TBT alternate mode in UFP port.	1: Enable TBT mode in UFP 0: Disable TBT mode in UFP	Not allowed ***Note:*** *USB-C Dock does not support the TBT alternate mode.*
CY_APP_DEBUG_LEVEL	Used to determine if a UART log needs to be transmitted or not.	0: Transmit critical log messages (log_level > LOGLEVEL_ERROR). 1: Transmit critical and error log messages (log_level > LOGLEVEL_WARNING). 2: Transmit critical, error, and warning log messages (log_level > LOGLEVEL_INFO). 3: Transmit all log messages	Allowed
CY_APP_LOG_DISCONNECT_EVT_ENABLE	Enable/disable option for logging disconnect event to internal flash.	1: Enable the disconnect event logging. 0: Disable the disconnect event logging.	Allowed
SYS_DEEPSLEEP_ENABLE	Enable/disable putting the PMG1-S3 device into a low-power mode when idle.	1: Enable Deep Sleep mode. 0: Disable Deep Sleep mode.	Allowed

Option	Description	Values	Change allowed
CY_APP_GET_REVISION_ENABLE	Enable/Disable option for transmission of Get revision message.	1: Enable Get Revision transmission. 0: Disable Get Revision transmission.	Allowed
CY_APP_USB_ENABLE	Enable/Disable option for USB FS interface for PMG1-S3.	1: Enable USB FS interface. 0: Disable USB FS interface.	Allowed
CY_APP_SMART_POWER_ENABLE	Enable/Disable support for smart power feature.	1: Enable smart power. 0: Disable smart power.	Allowed.
CY_APP_DMC_PHASE1_UPDATE_ENABLE	Enable/disable support for phase 1 of firmware update operation.	1: Enable phase 1 of firmware update 0: Disable phase 1 of firmware update	Allowed
CCGX_UPDATE	Enable/disable CCG7SC device firmware update.	1: Enable CCG7SC firmware update 0: Disable CCG7SC firmware update	Allowed
HX3_BOOT_WAIT_ENABLE	Enable/disable deferring I2C communication with CCG7c device on power-up since HX3 is using the I2C bus for communication with EEPROM.	'1' for deferring I2C communication with a CCGx device when multiple I2C masters are present on the same bus '0' for not deferring I2C communication with a CCGx device when multiple I2C masters are present on the same bus	Not allowed
CY_HPI_MASTER_ENABLE	Enable/disable option for HPI master	1: Enable HPI master support 0: Disable HPI master support	Allowed. ***Note:*** *If this option is disabled, then 'CCGX_UPDATE' should also be disabled.*
CY_APP_USB_HID_INTF_ENABLE	Enable/disable option for USB HID interface	1: Enable HID interface 0: Disable HID interface	Allowed

Option	Description	Values	Change allowed
ETAG_SUPPORT_ENABLE	Enable/Disable option for ETAG feature support.	1: Enable ETAG feature support 0: Disable ETAG feature support	Allowed
EXTENDED_ALERT_EVENTS_SUPPORT	Enable/disable option for extended alert events	1: Enable extended alert event support 0: Disable extended alert event support	Allowed
CY_APP_PD_ENABLE	Enable/disable option for USB PD support	1: Enable support for USB PD 0: Disable support for USB PD	Not allowed ***Note:*** *PD support should not be disabled in a Dock.*
PMG1_V5V_CHANGE_DETECT	Enable/disable option for detecting 5 V change	1: Enable detection of 5 V change 0: Disable detection of 5 V change	Allowed
CY_APP_BUCKBOOST_MP4247_ENABLE	Enable/disable option for MP4247 controller driver	1: Enable the driver for the MP4247 controller 0: Disable the driver for the MP2427 controller	Allowed
CY_APP_BUCKBOOST_RT6190_ENABLE	Enable/Disable option for RT6190 controller driver	1: Enable the driver for the RT6190 controller. 0: Disable the driver for the RT6190 controller.	Allowed
VBUS_RCP_ENABLE	Enable/disable detection and handling of reverse current faults	1: Enable VBUS RCP 0: Disable VBUS RCP	Allowed ***Note:*** *While the middleware libraries do support this feature, it is not supported by the PMG1-S3 Dock solution.*
VBUS_SCP_ENABLE	Enable/disable detection and handling of short circuit faults	1: Enable VBUS SCP 0: Disable VBUS SCP	Allowed
VBUS_OCP_ENABLE	Enable/disable detection and handling of VBUS	1: Enable OCP 0: Disable OCP	Allowed

Option	Description	Values	Change allowed
	Over Current faults		
VCONN_OCP_ENABLE	Enable/disable detection and handling of VConn Over-Current faults.	1: Enable VCONN OCP 0: Disable VCONN OCP	Allowed
VBUS_OVP_ENABLE	Enable flag for the internal comparator based Over Voltage Protection (OVP) scheme. Even if the OVP feature is enabled using this definition, it can be disabled at run-time using the configuration table.	1: Enable OVP 0: Disable OVP	Allowed
VBUS_UVP_ENABLE	Enable/disable detection and handling of under-voltage faults	1: Enable VBUS UVP 0: Disable VBUS UV.	Allowed
CY_APP_DEFER_SNK_VBUS_UVP_HANDLING	Enable/Disable option to defer UVP handling to ensure that a real UV fault occurred or a disconnect	1: Enable deferring UVP handling 0: Disable UVP handling	Allowed
CY_APP_FLASH_LOG_ENABLE	Enable/Disable option for flash log support	1: Enable flash log support 0: Disable flash log support	Allowed
DFP_ALT_MODE_SUPP	Enable/Disable option for alternate modes for DFP port	1: Enable support for alternate modes for DFP port 0: Disable support for alternate modes for DFP port	Allowed
UFP_ALT_MODE_SUPP	Enable/Disable option for alternate modes for UFP port	1: Enable support for alternate modes for UFP port 0: Disable support for alternate modes for UFP port	Allowed

Option	Description	Values	Change allowed
PMG1_HPD_RX_ENABLE	Enable/Disable option for HPD RX block of PMG1	1: Enable HPD RX block of PMG1 0: Disable HPD RX block of PMG1	Allowed
CY_HPD_ENABLE	Enable/Disable option for HPD block	1: Enable HPD block 0: Disable HPD block	Allowed
CCG_BB_ENABLE	Enable/Disable USB Billboard support.	1: Enable the Billboard interface 0: Disable Billboard support	Allowed
DP_DFP_SUPP	Enable/Disable option for display port alternate mode in DFP port.	1: Enable DP mode in DFP 0: Disable DP mode in DFP	Allowed
DP_UFP_SUPP	Enable/Disable option for display port alternate mode in UFP port.	1: Enable DP mode in UFP 0: Disable DP mode in UFP	Allowed
MINOR_SVDM_VER_SUPPORT	Enable/Disable option for minor SVDM version support.	1: Enable minor SVDM version support 0: Disable minor SVDM version support	Not allowed ***Note:*** *This is mandatory from PD spec revision 3.1, v1.6.*
BILLBOARD_1_2_2_SUPPORT	Enable/Disable option for supporting billboard specification revision 1.2.2	1: Enable Billboard specification revision 1.2.2 0: Disable Billboard specification revision 1.2.2	Allowed
CUSTOM_ALT_MODE_UFP_SUPP	Enable/Disable option for vendor-specific alternate mode support on upstream port	1: Enable vendor-specific alt mode support on US port 0: Disable vendor-specific altmode support on US port	Allowed
CUSTOM_ALT_MODE_DFP_SUPP	Enable/Disable option for vendor-specific alternate mode support on downstream port	1: Enable vendor-specific alt mode support on DS port 0: Disable vendor-specific altmode support on DS port	Allowed
CUSTOM_OBJ_POSITION	Enable/disable usage of VDM	1: Enable usage of VDM header object position	Allowed

Option	Description	Values	Change allowed
	header object position field for custom alt modes. **Note:** If this is disabled, then the object position field in the VDM header can be used for populating custom values	field for custom alt modes 0: Disable VDM header object position field for custom alt modes	
CY_APP_USBC_DOCK_EXCHANGE_CAP_DP_USB3	Enable/Disable option for exchange capabilities in USB-C Docks	1: Enable exchange capabilities in USB-C Docks 0: Disable exchange capabilities in USB-C Docks	Allowed

Do not change the following compile-time options (see Table 39) because the middleware libraries are built using the fixed values.

Table 39 Fixed compile time options in EZ-PD™ PMG1-S3 USB-C Dock Solution

Option	Description	Values used by middleware
CY_PD_SINK_ONLY	Enable/disable the port power role as sink only	0
CY_PD_SOURCE_ONLY	Enable/disable the port power role as source only	0
CY_PD_REV3_ENABLE	Enable/disable the USB PD specification Rev3 support	1
CY_PD_CBL_DISC_DISABLE	Enable/disable the cable discovery before the PD negotiation.	0
NO_OF_BC_PORTS	No of legacy battery charging ports supported	0
CY_PD_VCONN_DISABLE	Enable/disable VCONN supply.	0
CY_PD_USB4_SUPPORT_ENABLE	Enable/disable support for USB4 mode	1
CY_PD_EPR_ENABLE	Enable/disable USB PD EPR support	1
CY_PD_EPR_AVS_ENABLE	Enable/disable USB PD EPR AVS support	1
CY_PD_PPS_SRC_ENABLE	Enable/disable USB PD PPS support	1

6.1.3 HID interface

6.1.3.1 Custom report exchange

The USB HID interface is primarily used for custom report exchange between the USB host and the PMG1-S3 dock. For example, ETAG information between the host and PMG1-S3 Dock is done over the HID interface.

An open-source HID-based host application required for custom report exchange can be downloaded from the HID page.

The exchange of reports between the host and device can be done in any of the following two ways:

- **Control endpoints**: This is done by handling SET/GET report requests in a callback provided in the application layer.
- **Data endpoints**: This is done by handling the output reports received or input reports to be transmitted in hid_task().

6.1.3.1.1 Output reports

The output reports will contain custom commands that will be processed by the PMG1-S3 Dock. The endpoint 0x03 will be used to receive output reports. Table 40 provides the structure of an output report.

Table 40 Structure of output reports

Byte 0	Byte 1	Byte 2
Report ID	Command Opcode	Data

The Report ID is the method employed by USB HID to allow multiple report formats to be sent and received from the same interface. In the project, the report ID used for sending and receiving custom reports is HID_REPORT_ID_VENDOR (0x01).

The command opcode refers to the actual custom command that needs to be processed by the PMG1-S3 dock.

The data field will contain the additional data that is sent from the host. This will be used for processing the command by the PMG1-S3 Dock.

6.1.3.1.2 Input reports

After processing the received custom command, a response is sent to the host by PMG1-S3 dock as an input report. This response or input report will contain the status of the previously received custom command along with return data (if applicable). The endpoint 0x84 will be used for sending input reports. Table 41 provides the structure of an input report.

Table 41 Structure of input reports

Byte 0	Byte 1	Byte 2	Byte 3
Report ID	Command opcode	Command status	Return data if applicable for command opcode

In the project, the report ID used for sending and receiving custom reports is HID_REPORT_ID_VENDOR (0x01).

The command opcode refers to the actual custom command that was processed by the PMG1-S3 dock.

The command status refers to the status after processing the custom command, which was received as part of the previous output report. Table 42 lists the possible status codes that the PMG1-S3 Dock can return.

Table 42 Supported HID status codes

Command status	Value	Description
HID_STAT_SUCCESS	0x00	Success status
HID_STAT_CMD_NOT_SUPPORTED	0x01	Command not supported
HID_STAT_REPORT_ID_NOT_SUPPORTED	0x02	Report ID not supported
HID_STAT_ETAG_PROG_FAILED	0x40	Failure in programming ETAG
HID_STAT_ETAG_ALREADY_PROGRAMMED	0x41	ETAG already programmed
HID_STAT_ETAG_INVALID_DATA	0x42	Invalid ETAG DATA
HID_STAT_ETAG_FLASH_READ_FAILURE	0x43	Failure in reading ETAG from flash

The Return data refers to the optional data that will be sent to the host as part of the input report. This will be used only for those command opcodes that are used to retrieve data from the PMG1-S3 Dock.

Note:

1. *SET_REPORT request over control endpoint shall be followed by Get_REPORT request over control endpoint from the host to query the last command status and supporting data (if applicable).*
2. *Output report received over data EP must be followed by the transmission of input report over data EP, over which last command status and supporting data, if applicable, must be sent from the device to the host.*

6.1.3.1.3 Supported custom commands

Table 43 lists the commands that are supported by the EZ-PD™ PMG1-S3 USB-C Dock Solution in this version of Dock SDK:

Table 43 Supported HID commands

Command opcode	Description	Data in output report	Return data in input report
HID_READ_DOCK_STATUS	Used to read the dock status	N/A	Dock status
HID_READ_DOCK_COMPOSITE_VERSION	Used to read dock composite version	N/A	Dock composite version
HID_CMD_SET_ETAG_DATA	Used to store ETAG information	ETAG information that needs to be stored	N/A
HID_CMD_GET_ETAG_DATA	Used to retrieve ETAG information	N/A	Retrieved ETAG information

6.1.3.2 System control using HID interface

This feature can be used to sleep or wake up the USB host using the HID interface. The report ID HID_REPORT_ID_VENDOR_2 (0x02) is used for HID system control input reports. These input reports are sent through an interrupt IN endpoint (0x84).

System sleep is triggered by pressing POWER_BTN once. After the USB host goes to sleep, wake-up can be triggered by pressing POWER_BTN again.

6.1.3.3 Report descriptor update

The report descriptor used in the project is stored in the `gl_usb_hidReportDescriptors` array that can be found in the *usb_descr.c* source file. This array should be updated for modifying the report descriptor. The HID descriptor will be adjusted dynamically to populate the updated size of the report descriptor.

6.1.3.4 Adding support for new custom commands

The new custom commands can be added to support the user requirements. The HID reports are handled inside the `handle_output_report()` function that can be found in the *usb_hid.c* source file. Do the following to add the support for a new custom command:

1. Add the new command code to the enum “cy_en_usb_hid_command_code_t” defined in *usb_hid.h*.
2. Inside the `handle_output_report()` function, add a case for the new command code under the following switch statement:

```
switch(command_code)
```

3. Handle the user requirements inside the switch case.
4. Update the “command_status” with the operation status performed as part of handling the command.
5. If any data is to be sent to the host after processing the command, it should be stored in the “buffer” array. The “buffer” can only hold a maximum of 61 bytes. To store more data, the report descriptor should be modified to increase input report size.
6. Call the `prepare_input_report()` function.

6.1.4 ETAG update

Set the maximum size of ETAG_DATA using the “ETAG_DATA_SIZE” macro. In the design, this macro is set to 10 bytes. It can be varied up to 62 bytes without modifying report descriptors or up to 254 bytes by modifying report descriptors as per user requirement.

The ETAG_VALID field is updated by the firmware when the `HID_CMD_SET_ETAG_DATA` command is received for the very first time.

ETAG can only be programmed once. Therefore, the `HID_CMD_SET_ETAG_DATA` command can only be sent once to the PMG1-S3 dock, otherwise, it will result in the command to fail.

Note: Format the stored ETAG_DATA and then sent to the dock from the USB host. The firmware running on the PMG1-S3 Dock stores the received data as it is into the SPI metadata section of the SPI flash.

6.1.5 HPI master

The HPI master library provides a set of function callbacks for event handling. These function callbacks are used for notifying the device and port-specific events received from the HPI slave devices and any error condition detected by the HPI master library to the application layer. Users can add custom code within the registered callbacks to handle the events.

The following code snippet demonstrates the callback function, where custom event handling can be added.

```
bool hpi_master_event_handler(cy_hpi_master_context_t *context, cy_hpi_master_event_t *event)
{
    bool clearIntr = true;
    uint8_t event_log[2] = {0};

    if((event->port == CY_HPI_MASTER_PORT_NUMBER_0) || (event->port == CY_HPI_MASTER_PORT_NUMBER_1))
    {
        event_log[0]= 1 << (event->port + 1);
        event_log[1] = event->eventCode;

        /* Handle port specific events. */
        CY_APP_DEBUG_LOG(0, CY_APP_DEBUG_HPI_MASTER_EVT_RECEIVED, event_log, 2, CY_APP_DEBUG_LOGLEVEL_INFO, true);
    }
    else
    {
        clearIntr = ccg_handle_hpi_event(context, event);
    }

    return clearIntr;
}
```

Figure 32 Event handler callback

```
bool hpi_master_error_handler(cy_hpi_master_context_t *context, cy_hpi_master_event_t *event)
{
    bool retStatus = true;

    switch(event->eventCode)
    {
        case CY_HPI_MASTER_QUEUE_OVERFLOW:
            CY_APP_DEBUG_LOG(0, CY_APP_DEBUG_HPI_MASTER_QUEUE_PUSH_ERROR, NULL, 0, CY_APP_DEBUG_LOGLEVEL_CRITICAL, true);
            break;
        case CY_HPI_MASTER_I2C_FAILURE:
            break;
        default:
            break;
    }

    return retStatus;
}
```

Figure 33 Error handler callback

6.2 EZ-PD™ PMG1-S3 Griffin Creek Dock solution

6.2.1 Device configurator

Table 44 lists the various resources of PMG1-S3 used in the EZ-PD™ PMG1-S3 Griffin Creek Dock solution. The selection of some of these resources are fixed due to the capabilities of the PMG1-S3 device. The table also points out the changes allowed in the default design.

Table 44 Resources of PMG-1S3 used in EZ-PD™ PMG1-S3 Griffin Creek Dock solution

Peripheral	Pin	Name	Description	Changes allowed
GPIO	P1.0	DG_PM_S3_EN	S3 indication from Goshen ridge	It can be changed to a different pin
SWD	P1.1	CYBSP_SWCLK	SWD Clock for clocking the data	No changes are allowed
SWD	P1.2	CYBSP_SWDIO	SWDIO pin for data transfer	No changes are allowed
GPIO	P1.3	AR_INT_P1	Interrupt pin to Goshen ridge for PD Port 0	It can be changed to a different pin
GPIO	P1.4	GR_FORCE_PWR	Goshen ridge force power signal	It can be changed to a different pin.
SCB1 (I2C)	P1.5	RIDGE_SLAVE_SDA	Used for communication between PMG1-S3 and ridge	It can be changed to a different SCB
SCB1 (I2C)	P1.6	RIDGE_SLAE_SCL	Used for communication between PMG1-S3 and ridge	It can be changed to a different SCB
GPIO	P2.0	DG_PB_OVC1	Overcurrent indication to FL5801	It can be changed
SCB5 (I2C)	P2.2	I2CM_SDA	I2C communication interface for buck -boost controller and current sensor	It can be changed to a different SCB
SCB5 (I2C)	P2.3	I2CM_SCL	I2C communication interface for buck-boost controller and current sensor	It can be changed to a different SCB
GPIO	P2.5	LED_CTRL	This pin will be connected to an onboard LED. This pin will be used by the LED control module to control the LED	It can be changed to a different output pin to which an LED is connected
GPIO	P3.0	DG_PMG1_PTO	Power rail control for foxville	It can be changed
GPIO	P3.4	POWER_BTN	Used for sending “Button Press” and “Button Release” extended alert events	It can be changed to make use of a different input pin
GPIO	P3.5	LAN_WAKE_BTN	Button used for sending extended alert event “Controller Initiated Wake”	It can be changed to make use of a different input pin
SCB0 (I2C)	P4.0	GR_FXVL_I2C_SCL	I2C interface for communicating with Foxville	It can be changed to a different SCB

Peripheral	Pin	Name	Description	Changes allowed
			and Goshen ridge for firmware update	
SCB0 (I2C)	P4.1	GR_FXVL_I2C_SDA	I2C interface for communicating with foxville and Goshen ridge for firmware update	It can be changed to a different SCB
SCB2 (UART)	P5.0	CYBSP_DEBUG_UART_RX	UART receiver pin	It can be changed to a different SCB
SCB2 (UART)	P5.1	CYBSP_DEBUG_UART_TX	Used for transmitting UART debug logs	It can be changed to a different SCB
GPIO	P5.2	REG_EN	Buck boost controller enable pin	It can be changed to a different pin
GPIO	P5.4	PB_VBUS_EN	Port B FET control pin	It can be changed to a different pin
GPIO	P7.0	AR_INT_P2	Interrupt pin to Goshen ridge for PD Port 1	It can be changed to a different pin
SCB7 (SPI)	P7.3	FLASH_SPI_MISO	MISO line for the SPI interface used to communicate with external flash	It can be changed to a different SCB
SCB7 (SPI)	P7.4	FLASH_SPI_CLK	Clock line for the SPI interface used to communicate with external flash	It can be changed to a different SCB
SCB7 (SPI)	P7.5	FLASH_SPI_MOSI	MOSI line for SPI interface used to communicate with external flash	It can be changed to a different SCB
SCB7 (SPI)	P7.6	FLASH_SPI_CS	Chip select line for SPI interface used to communicate with external flash	It can be changed to a different SCB
USBFS	P8.0	USBDP	DP line for USB FS interface	No changes are allowed
USBFS	P8.1	USBDM	DM line for USB FS interface	No changes are allowed
PD Port 0	–	PD_PORT0	USB Power Delivery Port 0	No changes are allowed
PD Port 1	–	PD_PORT1	USB Power Delivery Port 1	No changes are allowed

Note:

1. *. Any modifications done to **Peripherals** and **Pins** tab of device configurator should be in accordance with the **Changes allowed** field in the table given above.*
2. *If Griffin Creek dock reference design is used as hardware, it is recommended not to change any resources mentioned in the table given above.*
3. *For more information on modifying the tabs of Device Configurator, see the Device Configurator user guide.*

6.2.2 Compile-time options

Table 45 Compile time options supported in EZ-PD™ PMG1-S3 Griffin Creek Dock solution

Option	Description	Values	Change allowed
`CY_APP_LED_CONTROL_ENABLE`	Enable/disable LED control module.	1: Enable the LED control module. 0: Disable the LED control module.	Allowed
`CY_USE_CONFIG_TABLE`	Enable/disable configuration table support	1: Enable the configuration table 0: Disable the configuration table	Not allowed *Note: Dock SDK makes use of the configuration table extensively.*
`CY_CONFIG_TABLE_TYPE`	Configuration table device type	'4' for dock configuration table	Not allowed *Note: This version of the dock SDK makes use of the config table type 4.*
`CY_APP_ROLE_PREFERENCE_ENABLE`	Enable/disable preference for attempting to switch to the port's preferred data and power roles	1: Enable preference for attempting to switch to the port's preferred data and power roles. 0: Disable preference for attempting to switch to the port's preferred data and power roles.	Allowed
`CY_APP_POWER_ROLE_PREFERENCE_ENABLE`	Enable/disable preference for port's power role.	1: Enable port power role preference.	Allowed

Option	Description	Values	Change allowed
		0: Disable port power role preference.	
CY_APP_PD_PDO_SEL_ALGO	PDO Selection algorithm **Note:** Applicable in sink role only	0: Pick the Source PDO, which delivers the maximum amount of power 1: Pick the Fixed Source PDO, which delivers the maximum amount of power. 2: Pick the Fixed Source PDO, which delivers the maximum current. 3: Pick the Fixed Source PDO, which delivers power at maximum voltage.	Allowed
CY_APP_UART_DEBUG_ENABLE	Enable/Disable option for UART logs	1: Enable support for UART logs 0: Disable support for UART logs	Allowed
CY_APP_RESET_ON_ERROR_ENABLE	Enable/Disable software reset on error	1: Enable software reset on error 0: Disable software reset on error	Allowed
CY_APP_WATCHDOG_HARDWARE_RESET_ENABLE	Enable/Disable option for watchdog reset	1: Enable watchdog reset 0: Disable watchdog reset	Allowed
CY_APP_FLASH_LOG_ROW_NUM	Flash address row number where the logs are stored.	0x3F5u	Allowed ***Note:*** *The user should make sure that the newly chosen row is from the reserved section of flash*
CY_APP_FLASH_LOG_BACKUP_ROW_NUM	Flash address row number where a redundant copy of the logs are stored	0x3F7u	Allowed *Note:* *The user should make sure that*

Option	Description	Values	Change allowed
			the newly chosen row is from the reserved section of flash
CY_APP_BOOT_LOADER_LAST_ROW	Last flash row number of the bootloader image space	0x0Fu	Not allowed
CY_APP_IMG1_LAST_FLASH_ROW_NUM	Last flash row number of FW1 Image	0x010Fu	Not allowe
CY_APP_IMG2_LAST_FLASH_ROW_NUM	Last flash row number of FW2 Image	0x3EFu	Not allowed
CY_APP_PD_USB4_SUPPORT_ENABLE	Enable/Disable USB4 Support feature at the application level	1: Enable USB4 support feature at the application level 0: Disable USB4 support feature at the application level	Allowed
CY_APP_DMC_ENABLE	Enable/Disable DMC support	1: Enable DMC support 0: Disable DMC support	Not allowed
BATTERY_CHARGING_ENABLE	Enable/Disable legacy battery charging support	1: Enable legacy charging support 0: Disable legacy charging support	Not allowed *Note:* *This feature is not supported in the PMG1-S3 dock solution*
VBUS_C_DISCHG_DS	Set VBus discharge drive strength	Value to be set from 1 to 16	Not allowed *Note:* *It is recommended not to change this option*
CY_APP_DEBUG_LEVEL	Used to determine if a UART log needs to be transmitted or not.	0: Transmit critical log messages (log_level > LOGLEVEL_ERROR). 1: Transmit critical and error log messages	Allowed

Option	Description	Values	Change allowed
		(log_level > LOGLEVEL_WARNING). 2: Transmit critical, error, and warning log messages (log_level > LOGLEVEL_INFO) 3: Transmit all log messages	
CY_APP_LOG_DISCONNECT_EVT_ENABLE	Enable/disable option for logging disconnect event to internal flash	1: Enable disconnect event logging 0: Disable disconnect event logging	Allowed
SYS_DEEPSLEEP_ENABLE	Enable/disable putting the PMG1-S3 device into a low-power mode when idle.	1: Enable Deep Sleep mode 0: Disable Deep Sleep mode	Allowed
CY_APP_GET_REVISION_ENABLE	Enable/Disable option for transmission of Get revision message	1: Enable Get Revision transmission 0: Disable Get Revision transmission	Allowed
CY_APP_USB_ENABLE	Enable/Disable option for USB FS interface for PMG1-S3.	1: Enable USB FS interface 0: Disable USB FS interface	Allowed
CY_APP_SMART_POWER_ENABLE	Enable/Disable support for smart power feature.	1: Enable smart power 0: Disable smart power	Allowed
CY_APP_DMC_PHASE1_UPDATE_ENABLE	Enable/Disable support for phase 1 of firmware update operation	1: Enable phase 1 of firmware update 0: Disable phase 1 of firmware update	Allowed
GR_FW_UPDATE_SUPPORT	Enable/Disable firmware update for Goshen Ridge.	1: Enable firmware update support 0: Disable firmware update support	Allowed

Option	Description	Values	Change allowed
FXVL_FW_UPDATE	Enable/Disable firmware update for Foxville	1: Enable firmware update support 0: Disable firmware update support	Allowed
EXTENDED_ALERT_EVENTS_SUPPORT	Enable/disable option for extended alert events.	1: Enable extended alert event support 0: Disable extended alert event support	Allowed
CY_APP_PD_ENABLE	Enable/Disable option for USB PD support.	1: Enable support for USB PD 0: Disable support for USB PD	Not allowed *Note: PD support should not be disabled in dock solutions.*
PMG1_V5V_CHANGE_DETECT	Enable/Disable option for detecting 5 V change.	1: Enable detection of 5 V change 0: Disable detection of 5 V change	Allowed
CY_APP_BUCKBOOST_MP4247_ENABLE	Enable/Disable option for MP4247 controller driver.	1: Enable the driver for the MP4247 controller 0: Disable the driver for MP2427 controller	Allowed
VBUS_RCP_ENABLE	Enable/disable detection and handling of reverse-current faults.	1: Enable VBUS RCP 0: Disable VBUS RCP	Allowed *Note: While the middleware libraries support this feature, it is not supported by the PMG1-S3 dock solution.*
VBUS_SCP_ENABLE	Enable/disable detection and handling of short-circuit faults	1: Enable VBUS SCP 0: Disable VBUS SCP	Allowed
VCONN_OCP_ENABLE	Enable/disable detection and handling of VCONN Overcurrent faults	1: Enable VCONN OCP 0: Disable VCONN OCP	Allowed

Option	Description	Values	Change allowed
VBUS_OCP_ENABLE	Enable/disable detection and handling of VBUS overcurrent faults.	1: Enable OCP 0: Disable OCP	Allowed
VBUS_OVP_ENABLE	Enable flag for the internal comparator-based overvoltage Protection (OVP) scheme. Even if the OVP feature is enabled using this definition, it can be disabled at run-time using the configuration table.	1: Enable OVP 0: Disable OVP	Allowed
VBUS_UVP_ENABLE	Enable/disable detection and handling of under-voltage faults	1: Enable VBUS UVP 0: Disable the VBUS UVP	Allowed
CY_APP_DEFER_SNK_VBUS_UVP_HANDLING	Enable/Disable option to defer the UVP handling to ensure that a real UV fault occurred or a disconnect	1: Enable the deferring UVP handling 0: Disable UVP handling	Allowed
CY_APP_FLASH_LOG_ENABLE	Enable/Disable option for flash log support	1: Enable flash log support 0: Disable flash log support	Allowed
MINOR_SVDM_VER_SUPPORT	Enable/Disable option for minor SVDM version support	1: Enable minor SVDM version support 0: Disable minor SVDM version support	Not allowed *Note: Mandatory from PD specification revision 3.1, v1.6.*
BILLBOARD_1_2_2_SUPPORT	Enable/Disable option for supporting billboard specification revision 1.2.2	1: Enable billboard specification revision 1.2.2 0: Disable billboard specification revision 1.2.2	Allowed

Do not change the following compile-time options (see Table 46) because the middleware libraries are built using the fixed values.

Table 46 Fixed compile time options used in EZ-PD™ PMG1-S3 Griffin Creek Dock Solution

Option	Description	Values used by middleware
CY_PD_SINK_ONLY	Enable/disable the port power role as sink only	0
CY_PD_SOURCE_ONLY	Enable/disable the port power role as source only	0
CY_PD_REV3_ENABLE	Enable/disable the USB PD Specification rev. 3 support	1
CY_PD_CBL_DISC_DISABLE	Enable/disable the cable discovery before the PD negotiation	0
NO_OF_BC_PORTS	No of legacy battery charging ports are supported	0
CY_PD_VCONN_DISABLE	Enable/disable the VCONN supply	0
CY_PD_USB4_SUPPORT_ENABLE	Enable/disable the support for USB4 mode	1
CY_PD_EPR_ENABLE	Enable/disable the USB PD EPR support	1
CY_PD_EPR_AVS_ENABLE	Enable/disable the USB PD EPR AVS support	1
CY_PD_PPS_SRC_ENABLE	Enable/disable the USB PD PPS support.	1
DFP_ALT_MODE_SUPP	Enable/disable the option for alternate modes for DFP port	1
UFP_ALT_MODE_SUPP	Enable/disable the option for alternate modes for UFP port	1
PMG1_HPD_RX_ENABLE	Enable/disable the option for HPD RX block of PMG1	1
CY_HPD_ENABLE	Enable/disable the HPD block option.	1
CCG_BB_ENABLE	Enable/disable the USB Billboard support.	1
TBT_DFP_SUPP	Enable/disable the option for TBT alternate mode in DFP port	1
TBT_UFP_SUPP	Enable/disable the option for TBT alternate mode in UFP port	1
DP_DFP_SUPP	Enable/disable the option for display port alternate mode in DFP port	1
DP_UFP_SUPP	Enable/disable the option for display port alternate mode in UFP port	1
RIDGE_SLAVE_ENABLE	Enable/disable the option for Ridge Slave Interface	1
VIRTUAL_HPD_ENABLE	Enable/disable the option for Virtual HPD usage	1
VIRTUAL_HPD_DOCK	Enable/disable virtual HPD handling for Intel-based Dock platforms	1
VPRO_WITH_USB4_MODE	Enable/disable the option for vPro with USB4 mode	1
GATKEX_CREEK	Enable/disable option for custom dock SoC-related functionality	1
MUX_DELAY_EN	Enable/disable the option to provide delay between data path updates for Intel-based systems	1
STORE_DETAILS_OF_HOST	Enable/disable the option for exchange capabilities in TBT docks	1

6.3 Firmware update

6.3.1 Adding new device to the device topology

Do the following to add or remove the device from the existing dock design:

1. Updating CDTT (contains the information about all the components connected to DMC) in DMC configuration (and other configuration parameters, if any) using the EZ-PD™ Dock Configuration Utility.
2. Once the configuration (Dock topology) is updated to include the new device, new device update support should also be added in the firmware as below:

   a) Configure the SCB block based on the interface (I2C, SPI, or UART) that the new device update happens using the Device Configurator in ModusToolbox™.

   b) Device module support: For any new device types other than DMC, supported CCGx, you must add the firmware module support to implement the new device's update. The new module should take care of the following:

   - Fill the "cy_stc_app_dmc_update_opern" structure with the function pointers implementing the FW update logic for the new device.
   - Add the source file implementing the firmware update logic functions under the *dmc* folder in the ModusToolbox™ solution structure.
   - Initialize the global device operation structure with the newly filled structure for the respective component ID in the `dmc_init_hw_interface ()` function.
   - Update the `init_dock_reset()` function for the newly added device in the CDTT.
3. Once the device module code support is added and integrated with the DMC solution module, rebuild the project to generate the DMC firmware binaries.

Note: Firmware update for the user-added devices must be developed based on the DMC flashing (update) module sample code. The same is shown in the below example using the DMC flashing (update) module.

Figure 34 shows the operation structure with function pointers for the DMC flashing module. "*dmc_flashing.c*" implements each of these functions as per the DMC flashing requirements.

```
/**
 * @brief Structure holding function pointers for DMC device modules.
 */
static const cy_stc_app_dmc_update_opern_t dmc_operation = {
    dmc_init_dev_param,
    NULL,
    NULL,
    dmc_prepare_update,
    dmc_flash_row_write,
    NULL,
    dmc_check_fw_version,
    dmc_jump_to_alternate,
    dmc_is_dev_query_deferred,
    dmc_update_logic,
    dmc_skip_jump_to_alt_request,
    NULL
};
```

Figure 34 Operation structure with function pointers for the DMC flashing module

Figure 35 shows the code snippet that initializes the global device operation structure in the `dmc_init_hw_interface ()` function for PMG1-S3 (DMC) device.

```
const cy_stc_app_dmc_update_opern_t **dev_opern = Cy_App_Dmc_GetDevUpdateOpern();

if (cdtt->signature == CY_APP_DMC_CDTT_VALID_SIG)
{
    dev_count = cdtt->dev_count;

    for (comp_id = 0; comp_id < dev_count; comp_id++)
    {
        topology = &cdtt->dev_info[comp_id];
        switch (topology->device_type)
        {
            case CY_APP_DMC_DEV_TYPE_DMC_PMG1S3:
                dev_opern[comp_id] = get_dmc_operation();
                break;
```

Figure 35 Initialization of global device operation structure

6.3.2 Module operation structure

The module operation structure “cy_stc_app_dmc_update_opern” is defined in *cy_app_dmc_common.h*.

The following are the function pointers in the “cy_stc_app_dmc_update_opern” structure:

- `init_dev_param`

  This function should query the respective device. Also, update the dock metadata RAM copy with the information related to the current running image, image validity, and firmware version of the image(s). `Cy_App_Dmc_SetRamImageStatus()` and `Cy_App_Dmc_UpdateRamVersions()` must update the information in the RAM copy.

  This function pointer cannot be left NULL in the operation structure of the device module.

  Reference code: `dmc_init_dev_param()` in *dmc_flashing.c*

- `init`

  This function should initialize the context variables of the firmware update, if any, is to be used during the firmware update. Invoke this function using the DMC state machine to before starting the firmware update to the device. If no initialization is needed, this function pointer can be left NULL.

- `deinit`

  This function should deinitialize the context variables of the firmware update, if any, being used during the firmware update. Invoke this function using the DMC state machine after the firmware update to the device is done and before starting the firmware update to another device.

  This function pointer can be left NULL in the operation structure of the device module, if no deinitialization is need to be done.

- `prepare_update`

  This function should prepare the device for the flashing operation. This should send any preparatory commands to the device before starting the image update to the device. Invoke this function using the DMC state machine before starting the actual image update.

  This function can be implemented in a blocking or non-blocking style, using the "*deferred" parameter.

  - Set *deferred = false, if the function is blocking.
  - Set *deferred = true, if it is non-blocking. In this case, the function callback passed as a parameter to this function should be invoked upon completing the function execution to indicate the completion of the same to the DMC state machine.

  This function pointer can be left NULL in the operation structure of the device module, if no preparatory commands need to be done.

  Reference code: `dmc_prepare_update()` in *dmc_flashing.c*

  This is an example for blocking an implementation with *deferred = false.

- `flash_row`

  This function should write the row data received from the DMC state machine to the device. Invoke this function using the DMC state machine for every row with the appropriate row number and row data along with it.

  This function can be implemented in a blocking or non-blocking style, using the "*deferred" parameter.

  - Set *deferred = false, if the function is blocking.
  - Set *deferred = true, if it is non-blocking. In this case, the function callback passed as a parameter to this function should be invoked upon completing the function execution to indicate the completion of the same to the DMC state machine.

  This function pointer cannot be left NULL in the operation structure of the device module.

  Reference code: `dmc_flash_row_write()` in *dmc_flashing.c*

  This is an example for blocking an implementation with *deferred = false

- `finish_update`

  This function should implement any finishing tasks that need to be performed for the device following the image update. Invoke this function using the DMC state machine after the last row of data is successfully written to the device or in case of any image write failure to the device. The "flashing_status" parameter that passed to the function indicates whether the image update to the device is completed successfully.

  This function is invoked after every image update to the device.

  This function can be implemented in a blocking or non-blocking style, using the "*deferred" parameter.

  - Set *deferred = false, if the function is blocking.
  - Set *deferred = true, if it is non-blocking. In this case, the function callback passed as a parameter to this function should be invoked upon completing the function execution to indicate the completion of the same to the DMC state machine.

  This function pointer can be left NULL in the operation structure of the device module, if no finish update tasks need to be done.

- `check_fw_version`

  This function should implement the logic for the firmware version check for restricting the image update conditionally based on the firmware version of the incoming new image and existing image in the device.

  This function pointer can be left NULL in the operation structure of the device module if no firmware version restriction need to be imposed on the device update.

  Reference code: `dmc_check_fw_version()` in *dmc_flashing.c*

- `jump_to_alternate`

  This function should implement the logic for initiating the jump to alternate image for the device, if needed. Otherwise, the function pointer can be kept NULL.

  Reference code: `dmc_jump_to_alternate()` in *dmc_flashing.c*

- `is_dev_query_deferred`

  If the device module needs more time for initialization, the `is_dev_query_deferred` function defers the calling of `init_dev_param`. If deferred, the `init_dev_param` will be called when the EZ-PD™ Dock Firmware Update tool is invoked to query the dock status before the firmware update.

  This function pointer can be left NULL in the device module operation structure if the device module needs no delay and init_dev_params call need not be deferred.

  Reference code: `dmc_is_dev_query_deferred()` in *dmc_flashing.c*

- `dev_update_logic`

  This function should implement the update logic specific for the device module based on the image mode of the device and the current running image.

  This function pointer cannot be left NULL in the device module operation structure. If left as NULL, then the device update will not be attempted by the DMC state machine

  Reference code: `dmc_update_logic()` in *dmc_flashing.c*

- `skip_jump_to_alt_request`

  This function should implement the logic specific for the device module, based on the image mode of the device and the current running image, before invoking the jump to an alternate request.

  This function pointer cannot be left NULL in the device module operation structure. If left as NULL, then jump to alternate request to the device update will not be attempted by the DMC state machine.

  Reference code: `dmc_skip_jump_to_alt_request()` in *dmc_flashing.c*

- `dev_config_hw_interface`

  This function should configure the hardware specific for the device module (such as the SCB and GPIOs needed for the device firmware update).

6.4 Debug module

6.4.1 New opcode support

Do the following for supporting a new opcode in the debug module:

1. Add the newly required opcodes to the end of enum “cy_en_debug_opcodes_t”.

2. Call the macro `CY_APP_DEBUG_LOG` at the location in the code where the debug logging needs to be done.

This will enable the UART logging and dynamic information logging into internal flash. Logging static information for a new opcode to internal flash is not possible directly as it requires the modification of the flash log library.

References

[1] See the following middlewares for the API guide, release notes, and quick start guide for each middleware library:

- PDStack
- PDUtils
- USBDEV
- HPI
- PdAltMode
- PmgAppCommon
- mtb-pdl-cat2 PDL

[2] This user guide should be read with the following Industry Standard Specifications:

- USB Power Delivery specification, revision 3.1, v1.8 (USB-IF)
- USB Type C Cable and Connector specification, Revision 2.2 (USB-IF)
- VESA DisplayPort Alt mode on USB Type-C Standard, Version 1.0 (VESA)
- VESA DisplayPort standard, v1.3 and 1.4 (VESA)
- Universal Serial Bus Device Class Definition for Billboard Devices, Revision 1.2.2 (USB-IF)
- Device Class Definition for Human Interface Devices (HID), 1.11 (USB-IF)

Glossary

AFC
Adaptive Fast Charging

APDO
Augmented Power Data Object

API
Application Programming Interface

AVS
Adjustable Voltage Supply

BC
Battery Charging

BMC
Bi-phase Manchester Code

BOD
Brown-Out Detect

CC
Configuration Channel

CDTT
Composite Dock Topology Table

CDP
Charging Downstream Port

CPU
Central Processing Unit

CRC
Cyclic Redundancy Check

DCP
Dedicated Charging Port

Glossary

DFP
Downstream Facing Port

DRP
Dual Role Port

EC
Embedded Controller

ECC
Elliptic Curve Cryptography

EEPROM
Electrically erasable programmable read-only memory

EMCA
Electronically Marked Cable Assemblies

EPR
Extended Power Range

FS
Full Speed

FWCT
Firmware Configuration Table

GNU
GNU is Not Unix (recursive acronym)

GPIO
General Purpose IO

HPI
Host Processor Interface

IDE
Integrated Development Environment

I2C
Inter-IC Communications interface

Glossary

INTR
Interrupt

ISR
Interrupt Service Routine

JTAG
Joint Test Action Group

KB
kilobyte, 1024 bytes

LSB
Least significant Bit

MSB
Most significant Bit

OCP
Overcurrent protection

OVP
Overvoltage protection

PD
Power Delivery

PDO
Power Delivery Object

PHY
Physical Layer

PPS
Programmable Power Supply

QC
Quick Charge

RSA
Rivest–Shamir–Adleman, public key cryptography algorithm

Glossary

SBU
Sideband Use. See Type-C specification

SCB
Serial Communication Block

SCP
Short-circuit protection

SDP
Standard Downstream Port

SHA
Secure Hash Algorithm, cryptographic hash function

SOP
Start of Packet symbol directed at USB-PD receptacle.

SOP'
Start of Packet symbol directed at USB-PD plug near the Downstream Facing Port (DFP)

SOP''
Start of Packet symbol directed at USB-PD plug near the Upstream Facing Port (UFP)

SOP*
Collectively denoting Start of Packet symbol (SOP, SOP' and SOP'')

SRAM
Static Random Access memory

SROM
Supervisory ROM

SVDM
Structured Vendor Defined Message

SWD
Serial Wire Debug

TBT
Thunderbolt. High speed peripheral connect technology from Intel.

Glossary

UFP
Upstream Facing Port

USB
Universal Serial Bus

USBFS
Full Speed Universal Serial Bus

USB PD
USB Power Delivery

UVP
Undervoltage protection

VDM
Vendor Defined Message

WDT
Watch Dog Timer

XRES
External Reset

Revision history

Document revision	Date	Description of changes
**	2024-02-19	Initial release
*A	2024-03-11	Updated metadata

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

Edition 2024-03-11

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about this document?

Email:

erratum@infineon.com

Document reference

002-39397 Rev. *A

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.

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.