Getting started with EZ-PD™ PMG1 MCU on ModusToolbox™ software

About this document

Scope and purpose

This application note introduces the capabilities of the EZ-PD™ PMG1 (Power Delivery Microcontroller Gen1) family of high-voltage microcontrollers (MCU) with USB-C Power Delivery (PD) and helps you to get started with your first project with PMG1 MCU in Eclipse IDE for ModusToolbox™ software.

Intended audience

This is primarily intended for developers using EZ-PD™ PMG1 MCU family on ModusToolbox™ software.

Associated part family

All EZ-PD™ PMG1 MCU devices.

Table of contents

1 Introduction

1.1 PMG1 MCU family general description and comparison

EZ-PD™ PMG1 (Power Delivery Microcontroller Gen1) is a family of high-voltage USB PD MCU. These chips include an Arm® Cortex® -M0/M0+ CPU and USB PD controller along with analog and digital peripherals. PMG1 MCU is targeted for embedded systems that provide/consume power to/from a high-voltage USB PD port and leverages the microcontroller to provide additional control capability.

Table 1 shows the comparison of features of different MCUs of the PMG1 MCU family. This table can be used to select the suitable PMG1 MCU for end application.

Table 1 Comparison of features of different MCUs of EZ-PD™ PMG1 MCU family

Subsystem or range	Item	PMG1-S0	PMG1-S1	PMG1-S2	PMG1-S3
CPU and memory subsystem	Core	Arm® Cortex®-M0	Arm® Cortex®-M0	Arm® Cortex®-M0	Arm® Cortex®-M0+
	Max. freq (MHz)	48	48	48	48
	Flash (KB)	64	128	128	256
	SRAM (KB)	8	12	8	32
Power Delivery	Power Delivery ports	1	1	1	1 port for 48-QFN 2 ports for 97-BGA
	Role	DRP	DRP	DRP	DRP
	MOSFET gate drivers	2x PFET	2x PFET	2x NFET	Flexible 2x NFET
	Fault protection	V_{BUS} OVP, UVP, and OCP. SCP (source configuration only)	V_{BUS} OVP, UVP, and OCP. SCP and RCP (source configuration only)	V_{BUS} OVP, UVP, and OCP.	V_{BUS} OVP, UVP, and OCP. SCP and RCP (source configuration only)
USB	Integrated Full-speed USB 2.0 device with Billboard Class support	No	No	Yes	Yes
Voltage range	Supply (V)	V_{DDD} (2.7 - 5.5) V_{BUS} (4 - 21.5)	V_{SYS} (2.75 - 5.5) V_{BUS} (4 - 21.5)	V_{SYS} (2.7 - 5.5) V_{BUS} (4 - 21.5)	V_{SYS} (2.8 - 5.5) V_{BUS} (4 - 28)
	I/O (V)	1.71 - 5.5	1. 71 - 5.5	1. 71 - 5.5	1. 71 - 5.5
Digital	SCB (configurable as I2C/UART/SPI)	2	4	4	7 for 48-QFN (out of which 5 can be configured as SPI and UART) 8 for 97-BGA
	TCPWM block (configurable as timer, counter or pulse width modulator)	4	2	4	7 for 48-QFN 8 for 97-BGA

Subsystem or range	Item	PMG1-S0	PMG1-S1	PMG1-S2	PMG1-S3
	Hardware authentication block (Crypto)	No	No	Yes (AES-128/192/256, SHA1, SHA2-224, SHA2-256, PRNG, CRC)	Yes (AES-128, SHA2-256, TRNG, vector unit)
Analog	ADC	2x 8-bit SAR	1x 8-bit SAR	2x 8-bit SAR	2x 8-bit SAR 1x 12-bit SAR
	On-chip temperature sensor	Yes	Yes	Yes	Yes
Direct Memory Access (DMA)	DMA	No	No	No	Yes
GPIO	Max # of I/O	12 (10 + 2 Fail-Safe)	17 (15 + 2 Fail-Safe)	20 (18 + 2 Fail-Safe)	26 (24 + 2 Fail-Safe) for 48-QFN 50 (48 + 2 Fail-Safe) for 97-BGA
Charging standards	Charging source	BC 1.2, AC, AFC, and QC 3.0	BC 1.2, AC	BC 1.2, AC	BC 1.2, AC, AFC, and QC 3.0
	Charging sink	BC 1.2, AC, and QC 2.0	BC 1.2, AC	BC 1.2, AC	BC 1.2, AC and QC 2.0
ESD protection	ESD protection	Yes (Human body model and charged device model)	Yes (Human body model and charged device model)	Yes (Human body model and charged device model)	Yes (Human body model and charged device model)
Packages	Package options	24-QFN (4x4 mm, 0.5 mm pitch)	40-QFN (6×6 mm, 0.5 mm pitch)	40-QFN (6 × 6 mm, 0.5 mm pitch) 42-CSP (2.63 × 3.18 mm, 0.4 mm pitch)	48-QFN (6x6 mm, 0.5 mm pitch) 97-BGA (6x6 mm, 0.5 mm and 0.65 mm pitch)

1.2 EZ-PD™ PMG1-S0 MCU

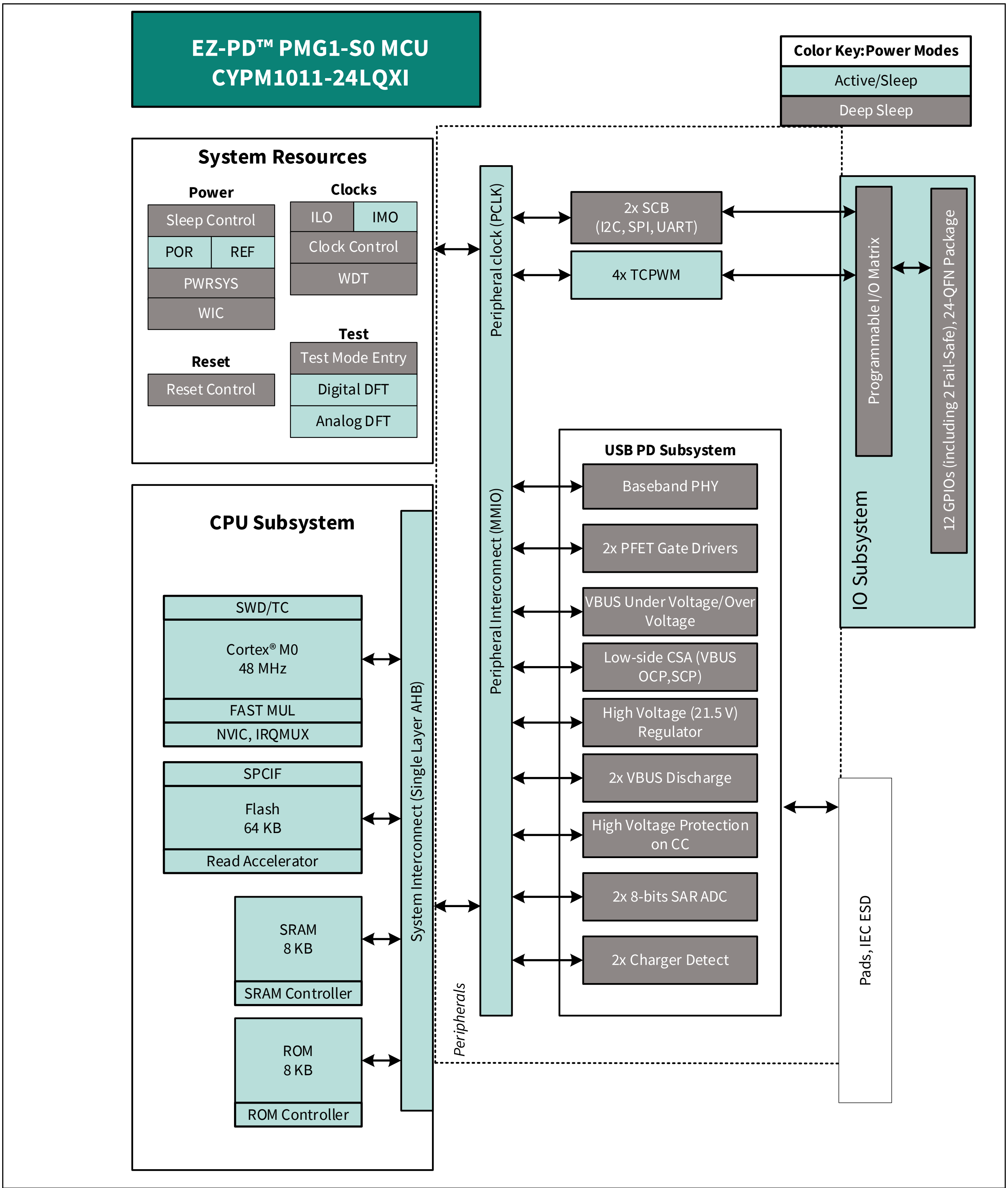


Figure 1 **PMG1-S0 MCU block diagram**

1.3 EZ-PD™ PMG1-S1 MCU

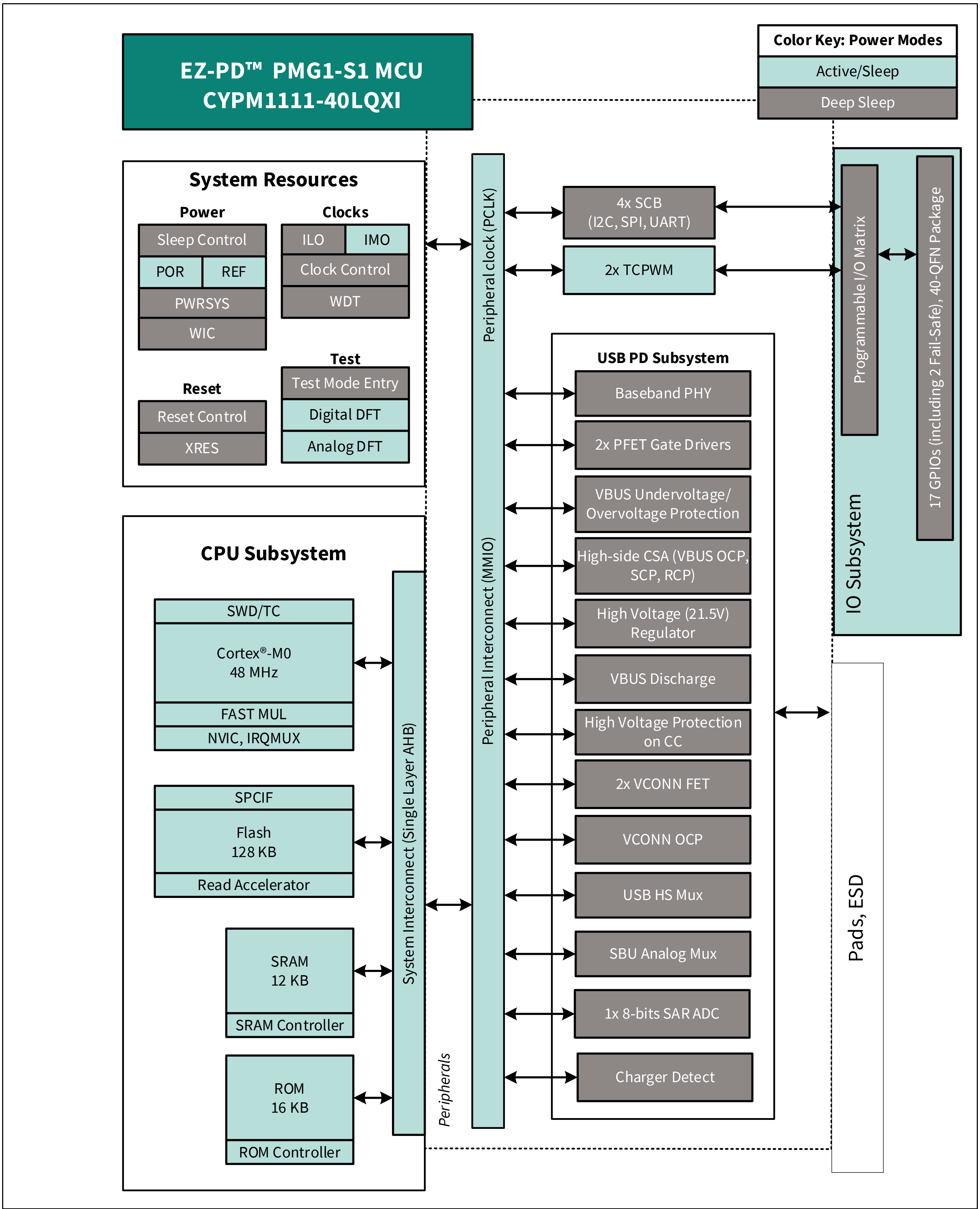


Figure 2 PMG1-S1 MCU block diagram

1.4 EZ-PD™ PMG1-S2 MCU

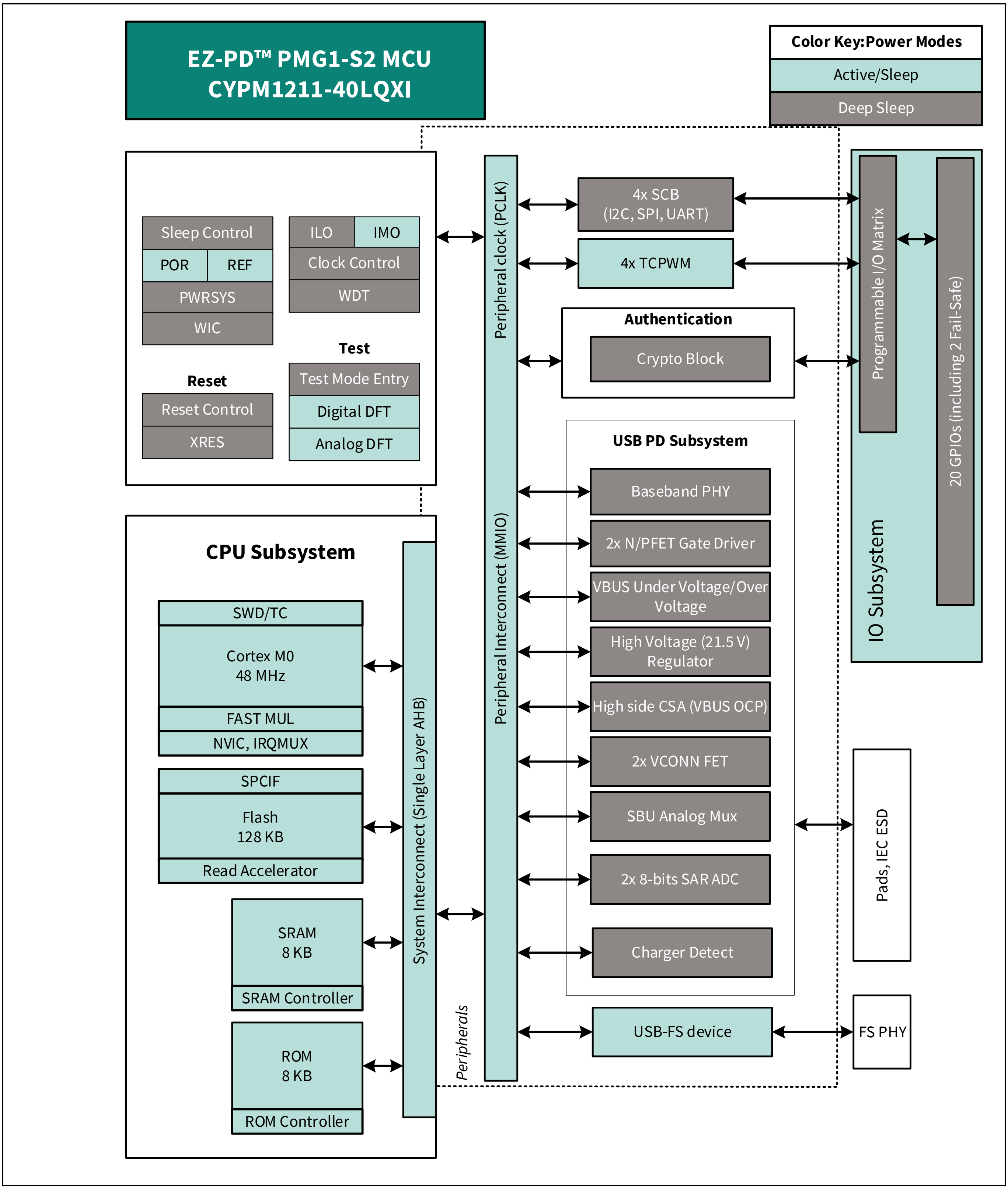


Figure 3 PMG1-S2 MCU block diagram

1.5 EZ-PD™ PMG1-S3 MCU

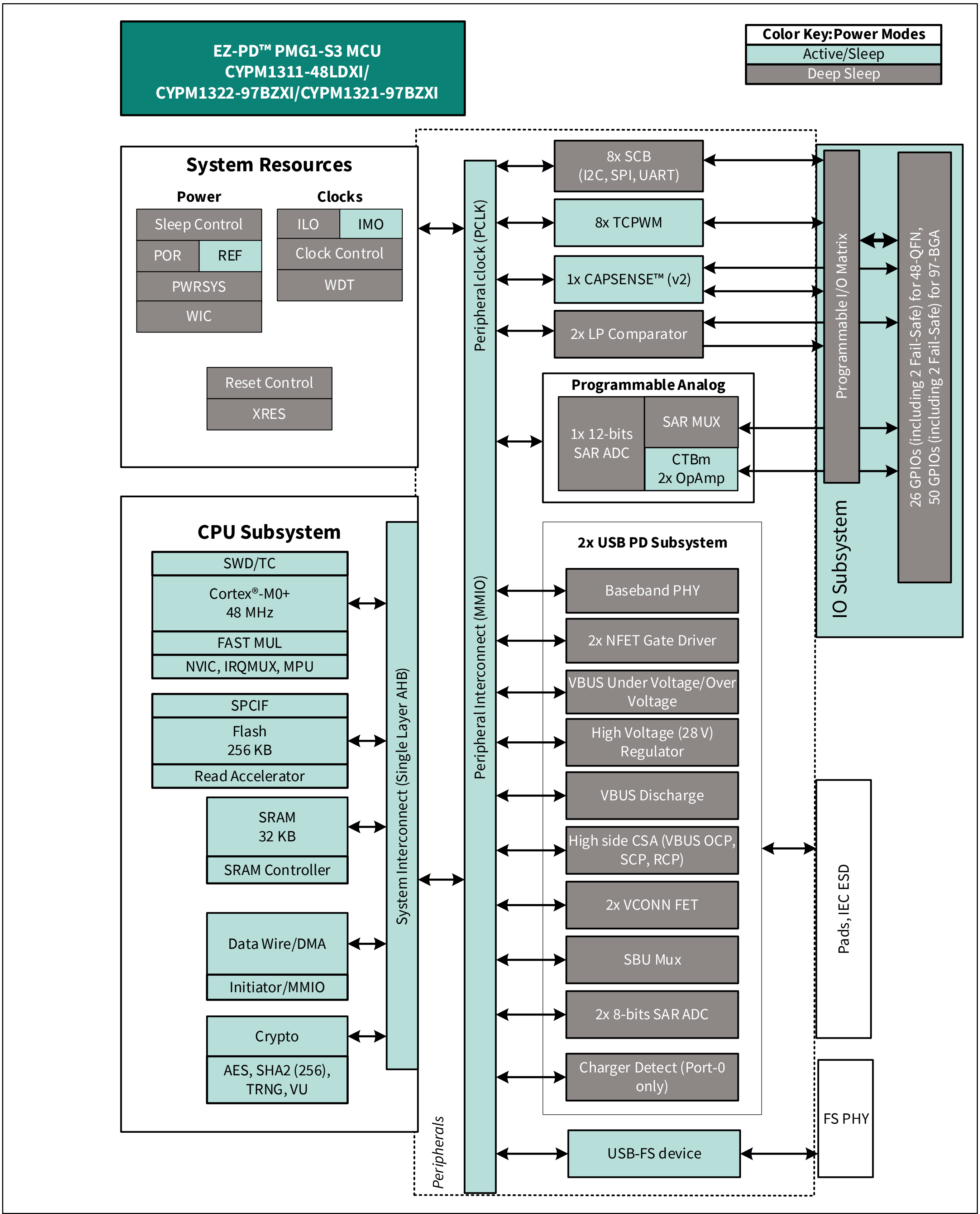


Figure 4 PMG1-S3 MCU block diagram

1.6 Software environment: ModusToolbox™ software

The ModusToolbox™ software environment supports PMG1 MCU application development with a set of tools for configuring the device, setting up peripherals, and complementing your projects with required middleware.

This application note gives an overview of the ModusToolbox™ development ecosystem and gets you started with a simple 'Hello World' application. The detailed steps to create the application from an empty template application are described in the following sections.

2 Development ecosystem

2.1 PMG1 MCU resources

The PMG1 MCU family has a rich set of documentation, development tools, and online resources to assist you during your development process. Visit PMG1 MCU webpage to find out more.

- Datasheets provide all the information needed to select and use the selected MCU, including functional description and electrical specifications.
- Application notes and code examples cover a broad range of topics, from basic to advanced level.
- Reference manuals provide detailed descriptions of the architecture and registers in each device family.
- Prototyping kits are available for evaluation, design, and development of different applications using PMG1 MCUs.
- Technical support: PMG1 MCU community forum, knowledge base articles.

2.2 ModusToolbox™ software

ModusToolbox™ development platform is used for firmware/application development with the PMG1 MCU. This latest-generation toolset includes the Eclipse IDE and therefore, is supported across Windows, Linux, and macOS platforms. The ModusToolbox™ software includes configurators, middleware libraries, as well as other packages that enable you to create your PMG1 MCU applications. Using the configurators, you can set the configuration of different blocks in the device and generate code that can be used in firmware development.

The Eclipse IDE for ModusToolbox™ is integrated with quick launchers for tools and design configurators in the Quick Panel. ModusToolbox™ also supports third-party IDEs, including Microsoft Visual Studio Code, Arm® MDK (µVision), and IAR Embedded Workbench.

A high-level view of the tools/resources included in the ModusToolbox™ software is shown in Figure 5. For a more in-depth overview of the ModusToolbox™ software, see ModusToolbox™ tools package user guide.

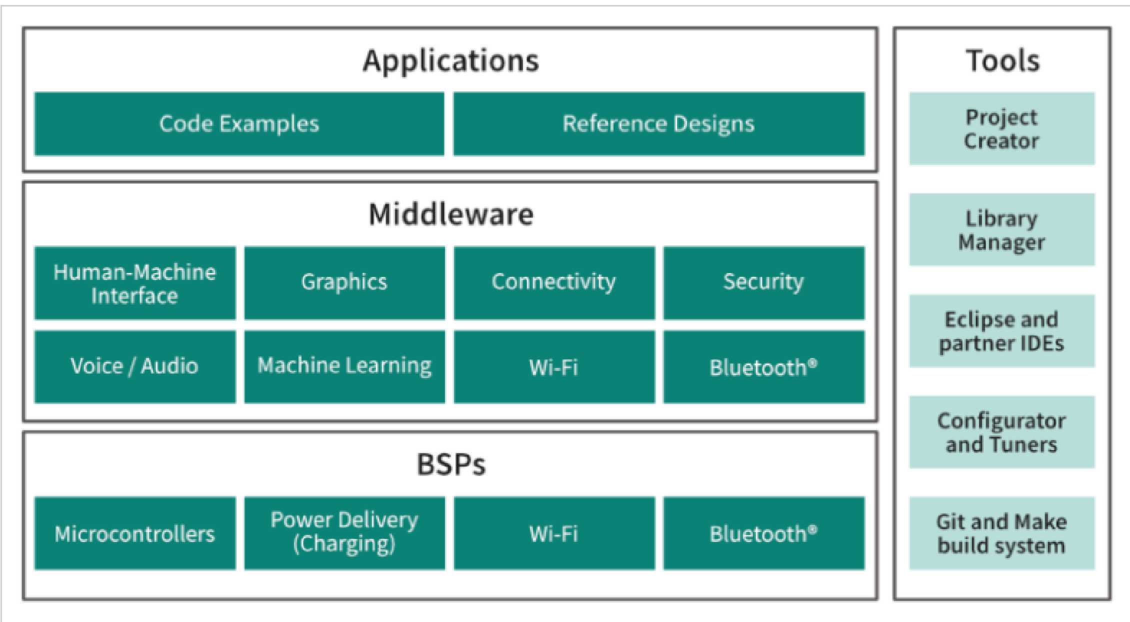


Figure 5 ModusToolbox™ software

2.3 PMG1 MCU software resources

The PMG1 MCU software resources include configurators, drivers, libraries, and middleware, as well as Makefiles, and scripts to get you started with developing firmware with PMG1 MCU.

2.3.1 Configurators

ModusToolbox™ software includes tools called Configurators that make it easier to configure a hardware block or middleware. For example, instead of having to search through all the documentation to configure a serial communication block as a UART with a desired configuration, launch Device Configurator and set the baud rate, parity, and stop bits. Upon saving the hardware configuration, the tool generates the “C” code to initialize the hardware with the desired configuration.

Configurators are independent of each other, but they can be used together to provide flexible configuration options. They can be used as standalone or in conjunction with other tools. Configurators are used for:

- Setting options and generating code to configure Peripheral drivers.
- Setting up connections such as pins and clocks for a peripheral.
- Setting options and generating code to configure middleware.

For PMG1 MCU applications, available configurators are as follows:

- **Device Configurator:** Used to enable and configure device peripherals, such as clocks and pins, as well as standard MCU peripherals that do not require their own configurator tool. This configurator generates initialization code used in the application.
- **EZ-PD™ Configurator:** Used for selecting the features and configuring parameters of the power delivery stack (PDStack) middleware for PMG1 family of devices. The tool generates configuration code in C language which can be referenced in the PDStack middleware.
- **CAPSENSE™ Configurator:** The CAPSENSE™ Configurator is included with ModusToolbox™ software, to create and configure CAPSENSE™ widgets, and generate code to interface the capacitive touch buttons and sliders.
- **USB Configurator**: The USB Configurator is used for configuring the USB data descriptors needed to implement USB 2.0 in application.

These configurators create their own files (e.g., *design.mtbezpd* for EZ-PD™ Configurator). The configurator file (*design.modus*) is usually provided with the BSP. When an application is created based on a BSP, the files are copied into the application. Developer can also create custom Device Configurator files for an application and override the ones provided by the BSP.

2.3.2 PMG1 MCU application development

ModusToolbox™ software tools and libraries come with source code and reference firmware enabling software development for PMG1 MCUs. This source code makes it easier to develop the firmware for supported devices. It helps the developer to quickly customize and build firmware without the need to understand the register set.

As Figure 6 shows, application development with ModusToolbox™ software:

1. Click on **Create New Application** to launch a popup window to choose a BSP (Project Creator).
2. Create a new application based on a list of template applications, filtered by kit. Now the firmware of the selected application is visible.
3. Open **Library Manager** and select **Add Library** to add middleware/software components, then click **OK**.

4. Start developing the application firmware using the PDL.

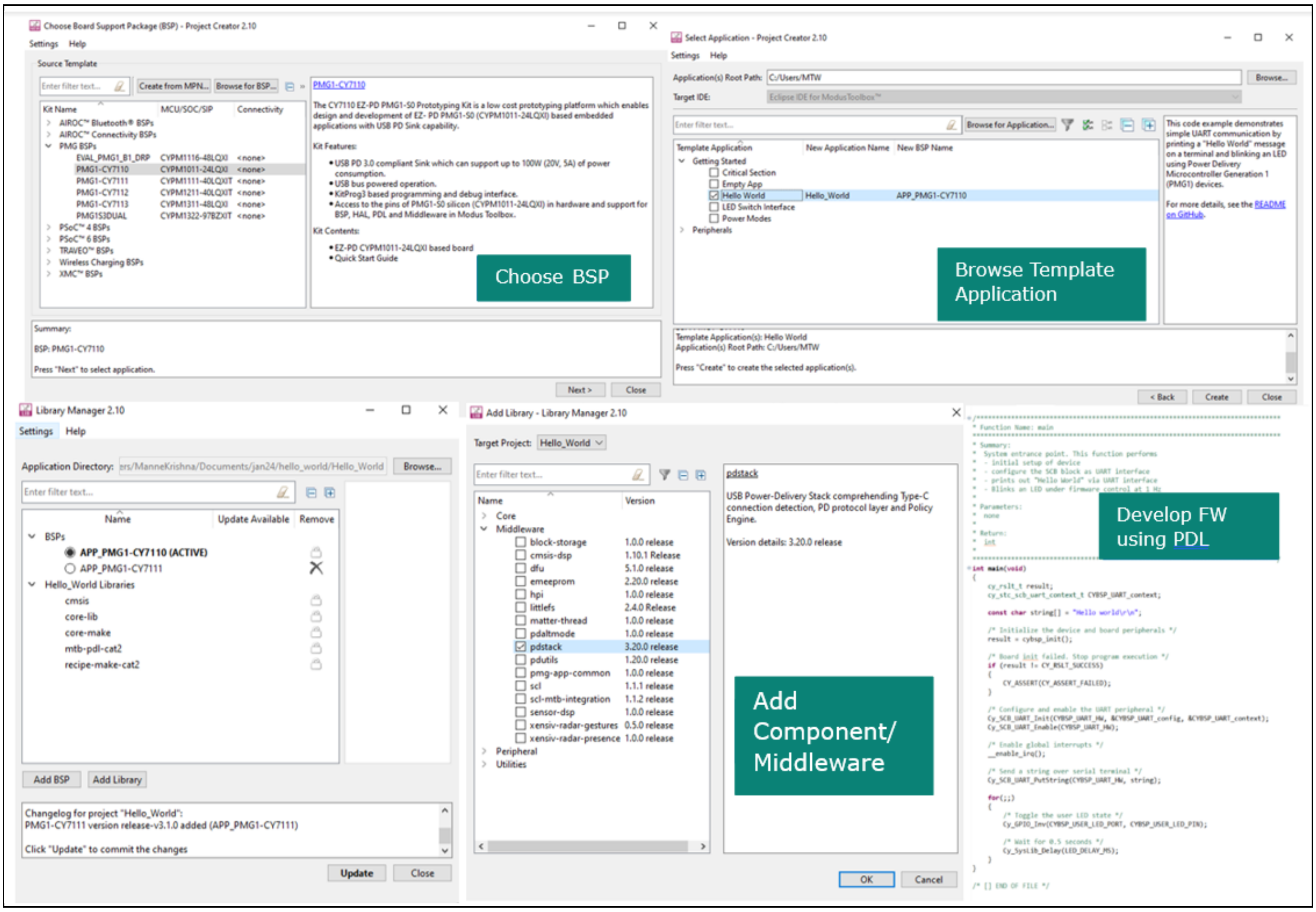


Figure 6 Eclipse IDE for ModusToolbox™ resources and middleware

2.4 ModusToolbox™ Help

Visit the ModusToolbox™ Software home page to download and install the latest version of ModusToolbox™ software. Launch Eclipse IDE for ModusToolbox™ software and navigate to the following items for ModusToolbox™ Help.

Choose **Help > Eclipse IDE for ModusToolbox™ General Documentation:**

- **ModulsToolbox™ documentation:** This page provides brief descriptions and links to various types of documentation included as part of ModusToolbox™ software.
- **User Guide:** Provides descriptions about creating applications as well as building, programming, and debugging them using Eclipse IDE
- **Installation guide :** This guide provides instructions for installing the ModusToolbox™ tools package.

2.5 Support for other IDEs

You can develop firmware for PMG1 MCUs using your favorite IDE such as IAR Embedded Workbench or Microsoft Visual Studio Code in addition to the Eclipse IDE.

See the "Exporting to IDE's" section in ModusToolbox™ tool package user guide for more details. Infineon recommends that you generate resource configurations using the configuration tools provided with ModusToolbox™ software.

2.6 Programming and debugging using Eclipse IDE

The Eclipse IDE of ModusToolbox™ software requires KitProg3 and uses the Open On-Chip Debugger (OpenOCD) protocol for debugging PMG1 MCU applications. It also supports GDB (GNU Debugger) debugging using industry standard probes like the Segger J-Link.

Note: *All PMG1 MCU kits have a KitProg3 onboard programmer/debugger. It supports Cortex® Microcontroller Software Interface Standard - Debug Access Port (CMSIS-DAP). See the KitProg3 user guide for details.*

For more information on programming/debugging firmware on PMG1 devices with ModusToolbox™ software, see the "Program and Debug" section in Eclipse IDE for ModusToolbox™ user guide.

2.7 Programming using mtb-programmer

ModusToolbox™ Programmer is a stand-alone, 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. For downloading visit mtb-programmer.

1. Make the following hardware connections on the prototyping kit before programming:
   a) Connect a Type-C cable from the J1 (KitProg3 USB Type-C port) connector on the kit to the host.
   b) Connect the J5 (power selection jumper) pins 2 and 3 for programming mode.

Note: *Device can also be programmed by connecting J5 (power selection jumper) pins 1 and 2 for operation mode and connecting both Type-C cables J1 and J10 to the host and power supply respectively.*

2. Select the device name (CY7110) in the Probe/Kit drop-down.
3. On the **Open Programming File** dialog, navigate to the location of the HEX, SREC, ELF, or BIN file to load, select it, and click **Open**.
4. Click **Connect**. ModusToolbox™ Programmer communicates with the device and displays various messages in the Log. Then, a message in the Status Bar indicates that it is connected as shown in Figure 7.
5. Click **Program**. ModusToolbox™ Programmer downloads the program file onto the device and displays messages in the Log. For detailed explanation on programming using mtb-programmer, see section 3 of ModusToolbox™ Programmer GUI user guide.

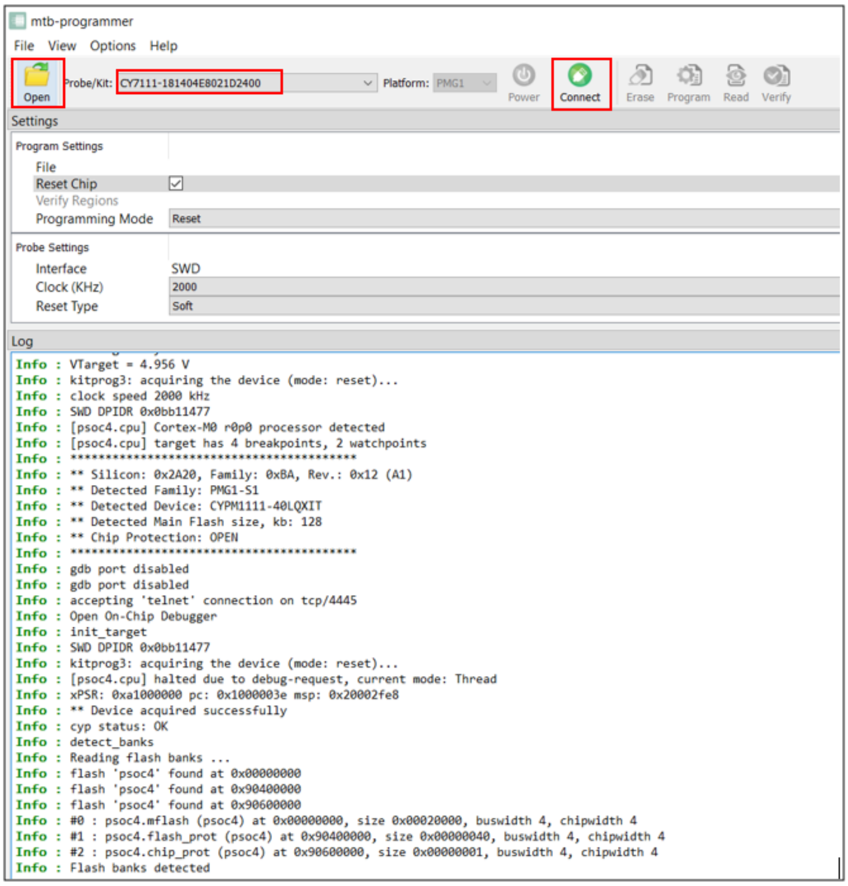


Figure 7 mtb-programmer tool

The mtb-programmer is available in the ModusToolbox™ software under **Quick Panel** after downloading it as shown in Figure 8. You can directly open it from there.

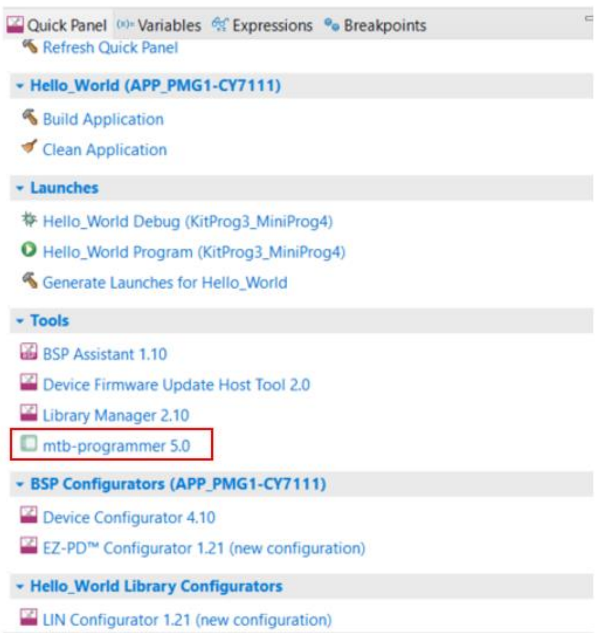


Figure 8 mtb-programmer tool under Quick Panel

2.8 PMG1 MCU kits

Table 2 BSPs for PMG1 MCU kits and corresponding BSPs

Sl no.	PMG1 MCU	PMG1 MCU Kit	BSP
1	CYPM1011-24LQXI	CY7110 EZ-PD™ PMG1-S0 Prototyping Kit	PMG1-CY7110
2	CYPM1111-40LQXI	CY7111 EZ-PD™ PMG1-S1 Prototyping Kit	PMG1-CY7111
3		EVAL_PMG1_S1_DRP	EVAL_PMG1_S1_DRP
4	CYPM1211-40LQXI	CY7112 EZ-PD™ PMG1-S2 Prototyping Kit	PMG1-CY7112
5	CYPM1211-42-CSP	NA	PMG1-CY7112
6	CYPM1311-48LQXI	CY7113 EZ-PD™ PMG1-S3 Prototyping Kit	PMG1-CY7113
7	CYPM1322-97BZXI	NA	PMG1S3DUAL
8	CYPM1321-97BZXI	EVAL_PMG1_S3_DUALDRP	EVAL_PMG1_S3_DUALDRP

3 Creating a PMG1 MCU application using Eclipse IDE for ModusToolbox™ software

This introduces how to use ModusToolbox™ software for project creation, programming, and debugging of EZ-PD™ PMG1 MCUs.

3.1 Prerequisites

- Select appropriate kit from PMG1 MCU product line.
- Download latest ModusToolbox™ software from ModusToolbox™ webpage.

After installing the software, see the ModusToolbox™ tools package user guide to get an overview of the software. You also need internet access to get the GitHub repositories during project creation.

See Table 2 for the list of PMG1 MCU kits.

3.2 Create a new application

1. Open Eclipse IDE for ModusToolbox™.
2. Navigate to **Quick Panel** and click **New Application** under **Start**. Alternatively, you can choose **File** > **New** > **ModusToolbox™ Application**, as shown in Figure 9.

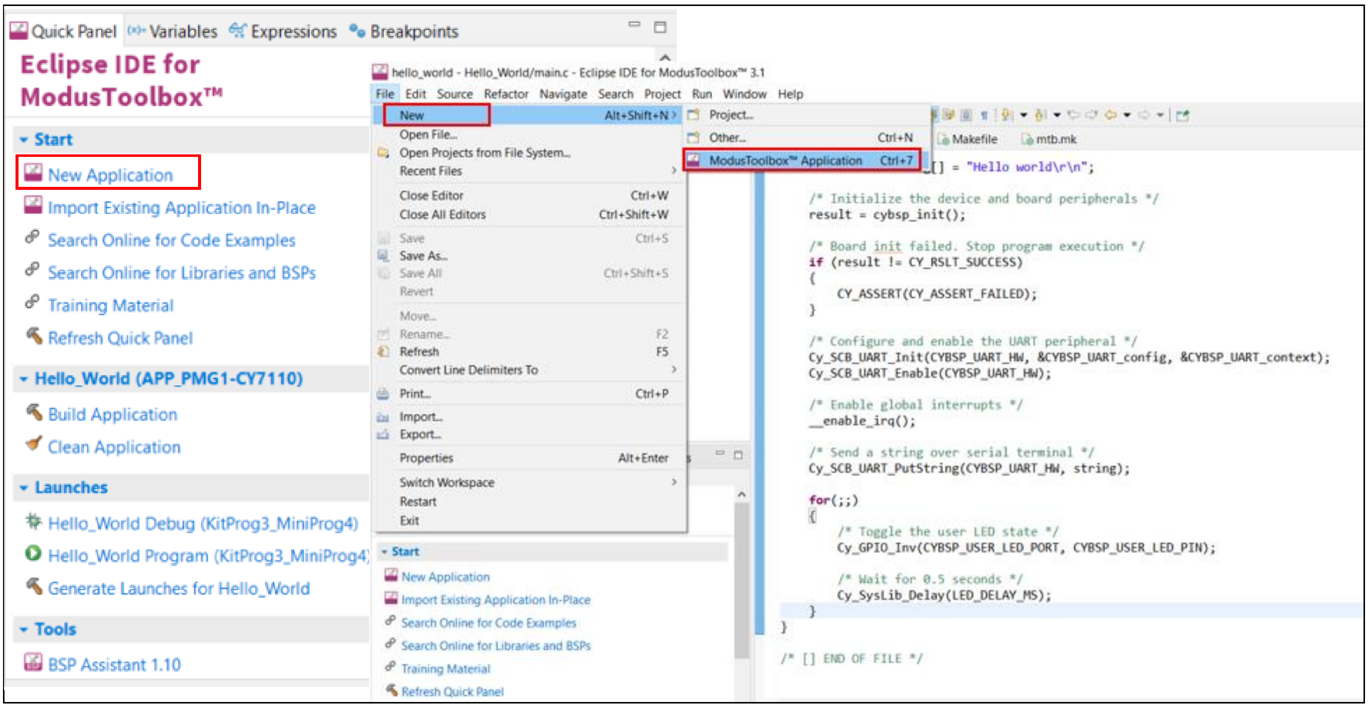


Figure 9 Create a new ModusToolbox™ application

3. In the Choose Board Support Package (BSP) dialog, choose the kit name. For example, for PMG1-S0, choose PMG1-CY7110. See Figure 10.
4. Click **Next**.

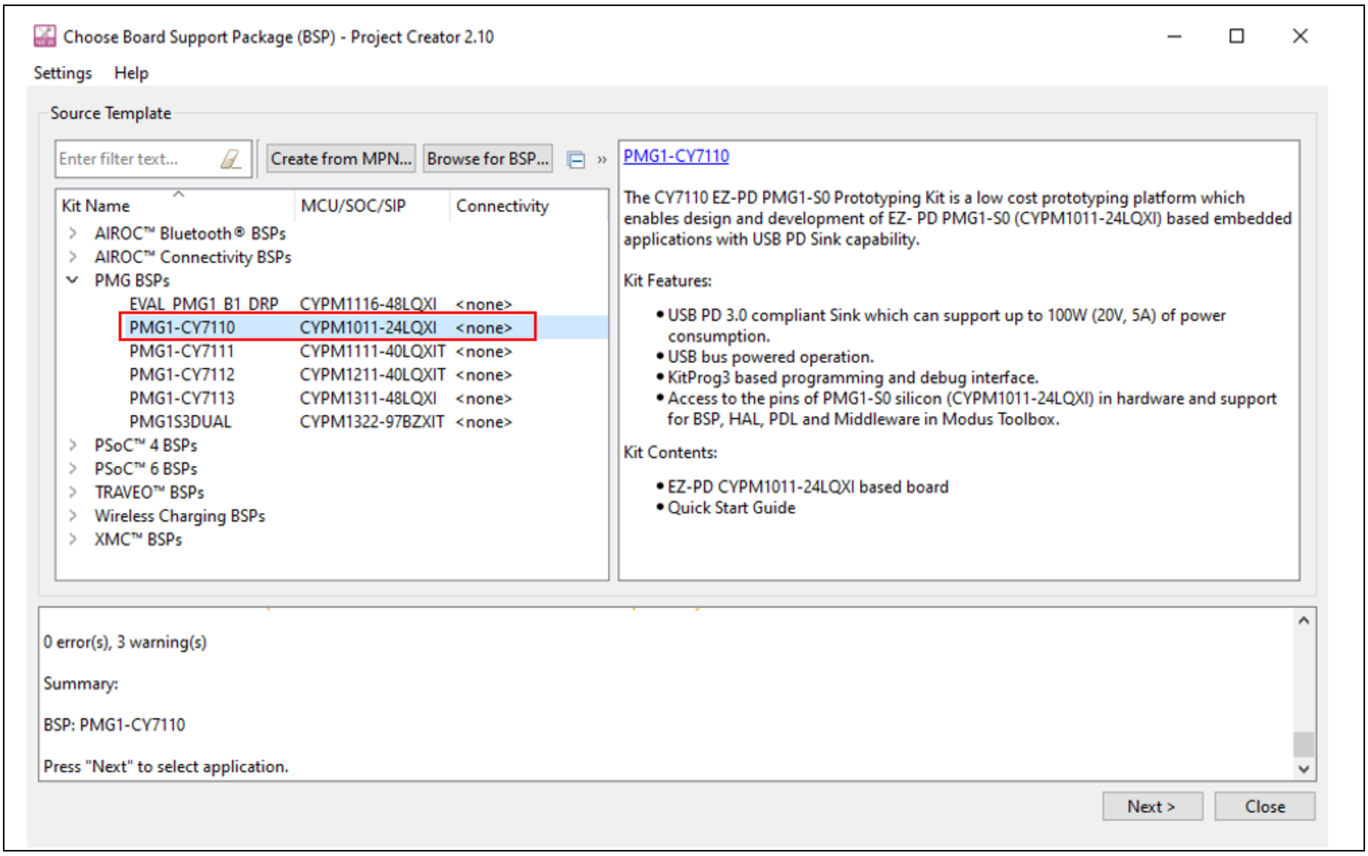


Figure 10 Choose the target hardware

5. In the **Select Application** dialog, select Hello World template application, as Figure 11 shows.
6. In the **Name** field, type in a name for the application, such as PMG1_S0_Hello_World. Alternatively, you can choose to leave the default name.
7. In the application(s) Root Path enter the location of workspace where you want to store the application Firmware.
8. Click **Create** to create the application.
9. The **Project Creator** will automatically close once the project is successfully created.

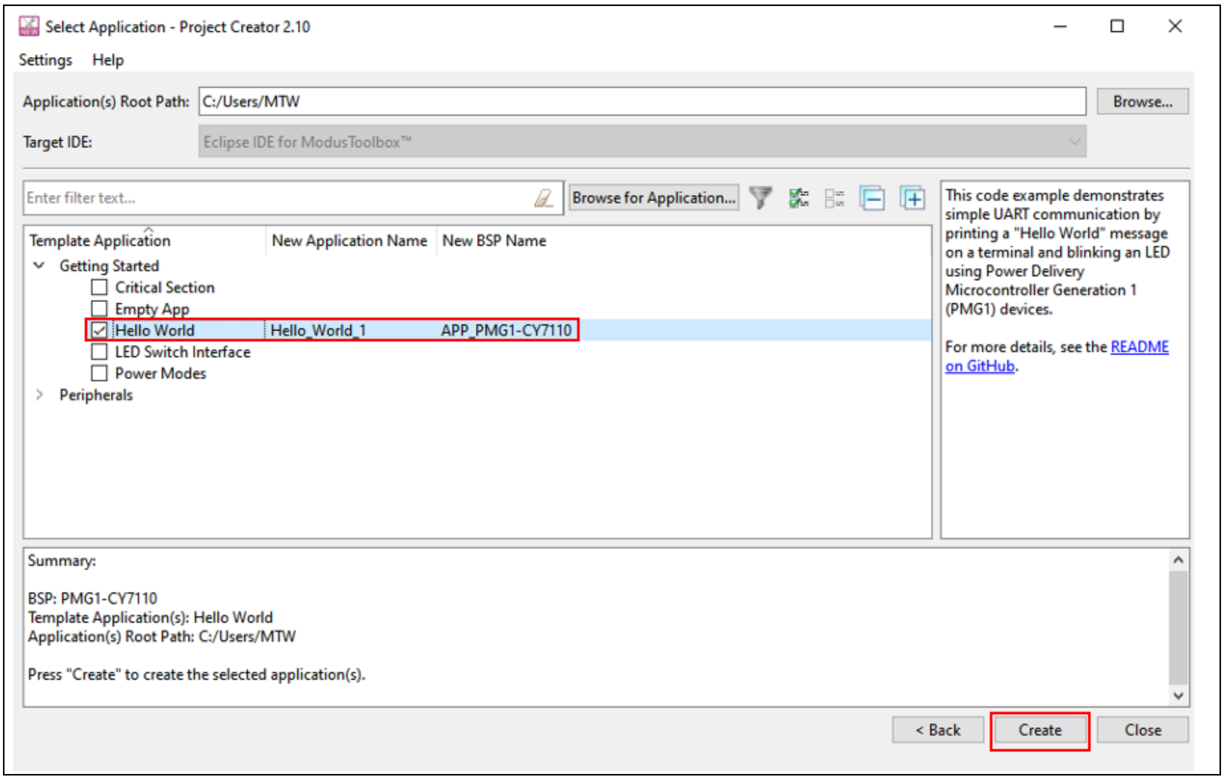


Figure 11 Choose starter application

You have successfully created a new ModusToolbox™ software application for PMG1-S0 MCU. The BSP uses the CYPM1011-24LQXI device on the CY7110 EZ-PD™ PMG1-S0 Prototyping Kit (PMG1-CY7110).

README.md file will open after the application is created successfully, which contains information about the template application implementation.

If you are using custom hardware based on PMG1-S0 MCU, or a different PMG1 MCU part number, see the "Creating/edit BSP" section in the ModusToolbox™ user guide (**Help** > **ModusToolbox™ General Documentation> ModusToolbox™ user guide**).

3.3 View and modify the design configuration.

Figure 12 shows the Eclipse IDE Project Explorer interface displaying the structure of the application project.

In the Eclipse IDE for ModusToolbox™ software, a PMG1 MCU application consists of a project to develop code for the CM0/CM0+ CPU. A project folder consists of various subfolders – each denoting a specific aspect of the project.

- An application project contains a Makefile which is typically at the root folder. It has instructions on how to recreate the project. There can be more than one project in an application; each dependent project usually resides within its own folder within the application folder and contains its own Makefile.
- The build folder contains all the artifacts resulting from the make build of the project. The output files are organized by target BSPs.

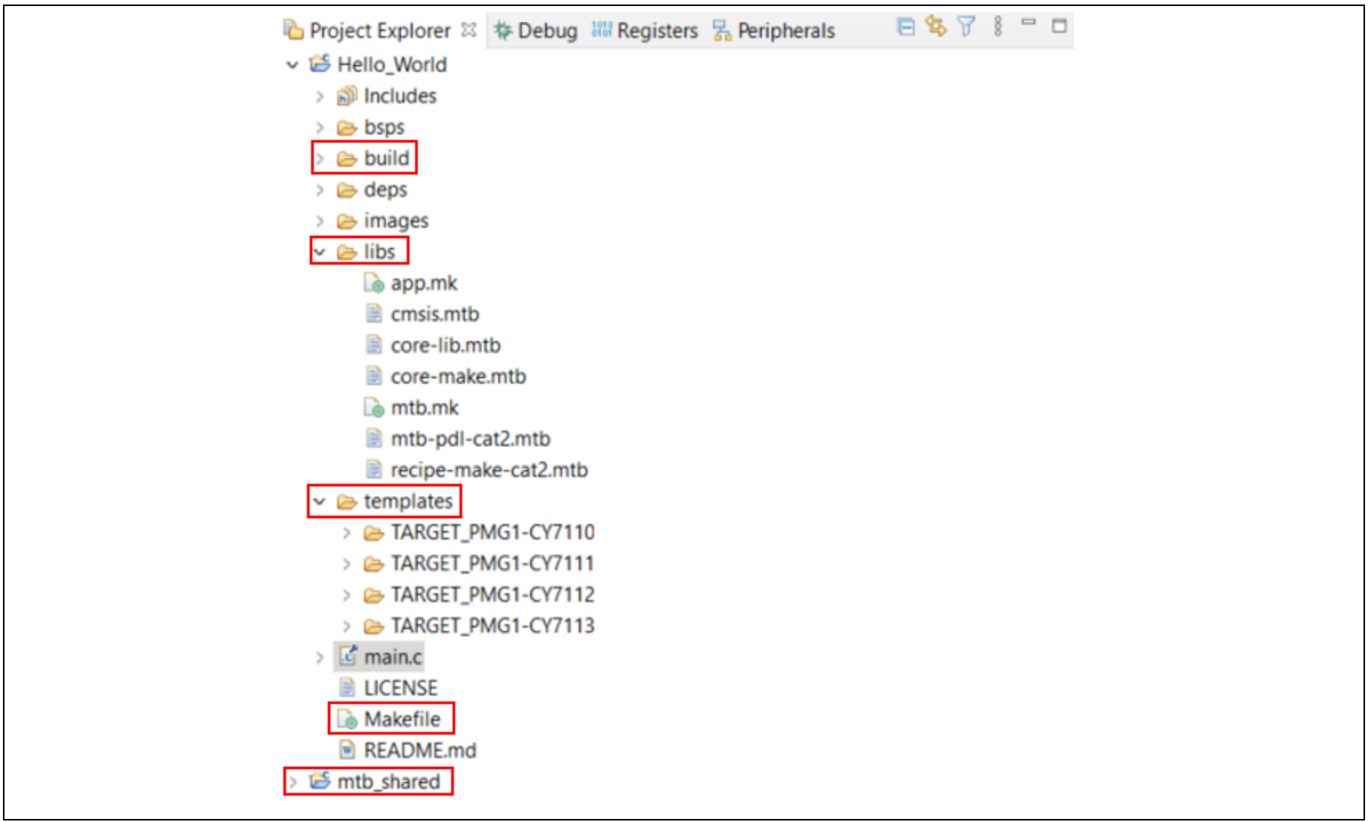


Figure 12 Project Explorer view

- The libs folder contains *mtb.mk* file which stores the relative paths of the all the libraries required by the application. The build system uses this file to find all the libraries required by the application.
- By default, when creating a new application or adding a library to an existing application and specifying it as shared, all libraries are placed in an *mtb_shared* directory adjacent to the application directory. The *mtb_shared* folder is shared between different applications that use the same versions of BSP/library.

3.4 Opening the Device Configurator

The template folder contains target files and has the *design.modus* file.

1. Click the *design.modus* file to open the design configuration as defined by the BSP.
2. Click the **Device Configurator** link in **Quick Panel** to view and modify the design configuration. See Figure 13.

You can also double-click to open the other *design.modus* files under the template section to open the respective configurators or click the corresponding links in **Quick Panel**.

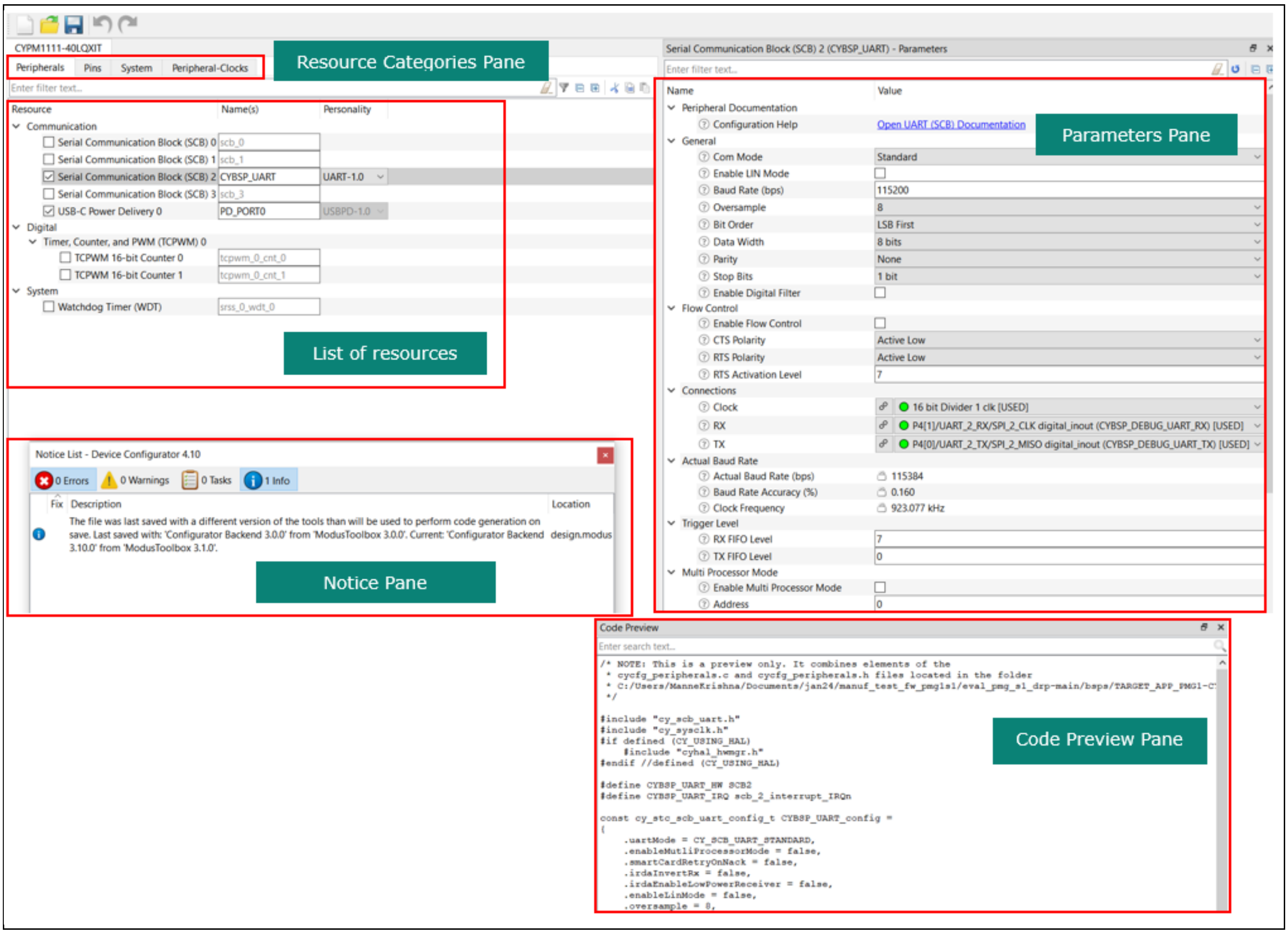


Figure 13 ***design.modus* overview**

3. From **List of Resources** in the **Resources Categories** tab in **Device Configurator**, choose from the different resources available in the device such as peripherals, pins, and clocks.

   You can choose how a resource behaves by choosing a Personality for the resource. For example, a Serial Communication Block (SCB) resource can have a EZI2C, I2C, SPI, or UART personalities. The Alias is the name for the resource, which is used in firmware development. One or more names can be specified by using a comma to separate them (with no spaces).

4. In the **Parameters** pane, enter the configuration parameters for each enabled resource and the selected personality.

   The **Code Preview** pane shows the configuration code generated per the configuration parameters selected. This code is populated in the *cycfg_* files in the *GeneratedSource* folder.

   The **Notices** pane displays the errors, warnings, and information messages arising out of the configuration.

   Developer can use API calls provided by PDL to initialize and use the peripheral. See “Configure and enable the UART peripheral” part in the main.c (Hello World code example) for more information. See Figure 14.

3.5 Write the firmware

At this point in the development process, you have created an application, with the assistance of a template application.

- If you are using the PMG1 MCU 'Hello World' code example, all the required files are already in the application. One can alter this application according to their requirement.
- If you want to create your own application from the scratch, choose the "**Empty App**" available under "**Select Application**" and modify the *main.c* file and device configurator accordingly, to achieve the required functionality. See Figure 11 for Empty application.

3.5.1 Firmware flow

- Examine the code in the *main.c* file of the application. For the firmware flowchart, See Figure 14.
- Resource initialization for this example is performed by the CM0 CPU. It configures the system clocks, pins, clock to peripheral connections, and other platform resources.
- When the CM0 CPU is enabled, the clocks and system resources are initialized by the BSP initialization function. The UART peripheral is configured and enabled. It prints a "Hello World" message on the terminal emulator. The CPU continuously toggles the LED state with a delay of 500 milliseconds.

Note: *Note that the application code uses BSP/middleware functions to execute the intended functionality.*

```
/*******************************************************************************
 * Include header files
 ******************************************************************************/
#include "cy_pdl.h"
#include "cybsp.h"


/*******************************************************************************
* Macros
*******************************************************************************/
#define LED_DELAY_MS              (500u)
#define CY_ASSERT_FAILED          (0u)


* Function Name: main
int main(void)
{
    cy_rslt_t result;
    cy_stc_scb_uart_context_t CYBSP_UART_context;

    /* Initialize the device and board peripherals */
    result = cybsp_init();

    /* Board init failed. Stop program execution */
    if (result != CY_RSLT_SUCCESS)
    {
        CY_ASSERT(CY_ASSERT_FAILED);
    }

    /* Configure and enable the UART peripheral */
    Cy_SCB_UART_Init(CYBSP_UART_HW, &CYBSP_UART_config, &CYBSP_UART_context);
    Cy_SCB_UART_Enable(CYBSP_UART_HW);

    /* Enable global interrupts */
    __enable_irq();

    /* Send a string over serial terminal */
    Cy_SCB_UART_PutString(CYBSP_UART_HW, "Hello world\r\n");

    for(;;)
    {
        /* Toggle the user LED state */
        Cy_GPIO_Inv(CYBSP_FW_LED_PORT, CYBSP_FW_LED_PIN);

        /* Wait for 0.5 seconds */
        Cy_SysLib_Delay(LED_DELAY_MS);
    }
}

/* [] END OF FILE */
```

Figure 14 main.c

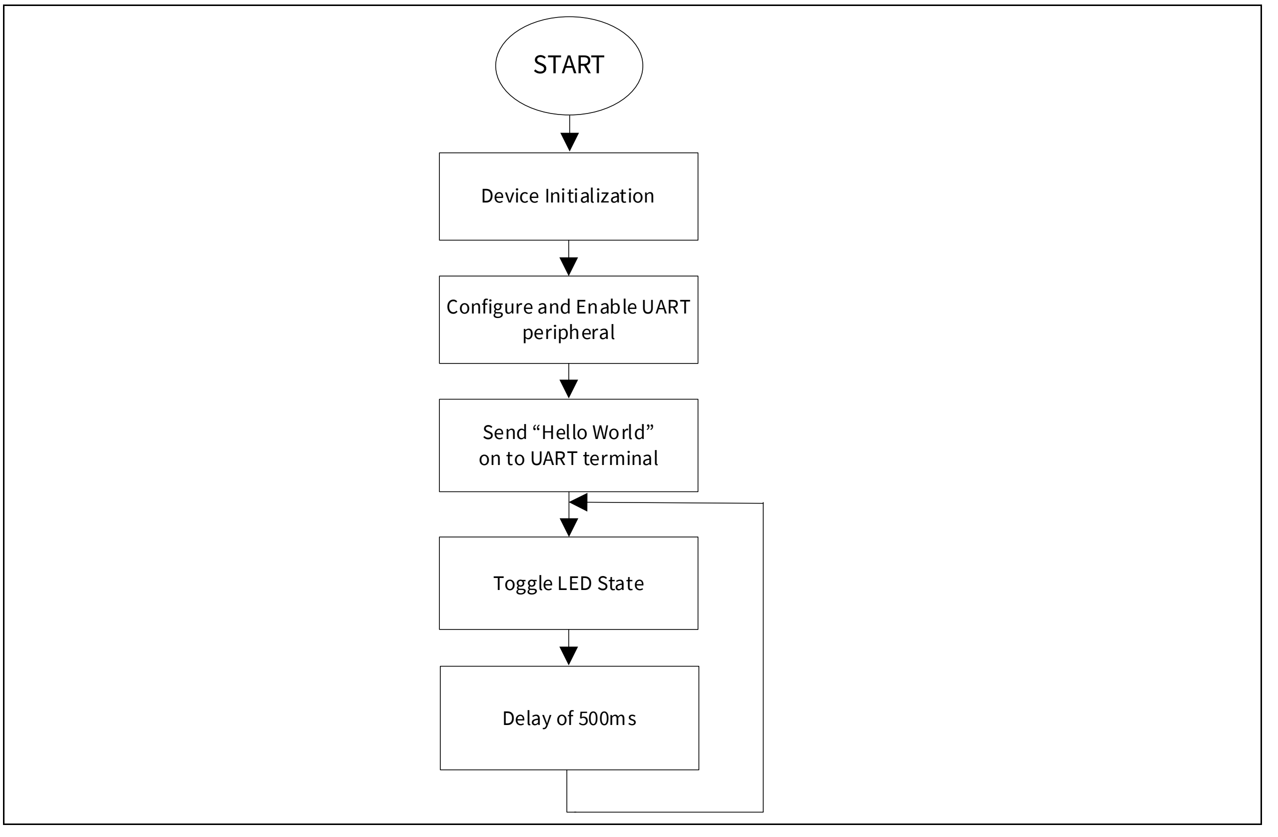


Figure 15 Firmware flowchart

This completes the summary of how the firmware works in the code example.

3.6 Build and program the application

Building the program.

1. Select the application project in the Project Explorer window and click on the **Build Application** shortcut under the **<application name>** group in **Quick Panel**. It selects the **Debug** build configuration and compiles/links all projects that constitute the application.
2. The **Console** view lists the results of the build operation, as Figure 16 shows.

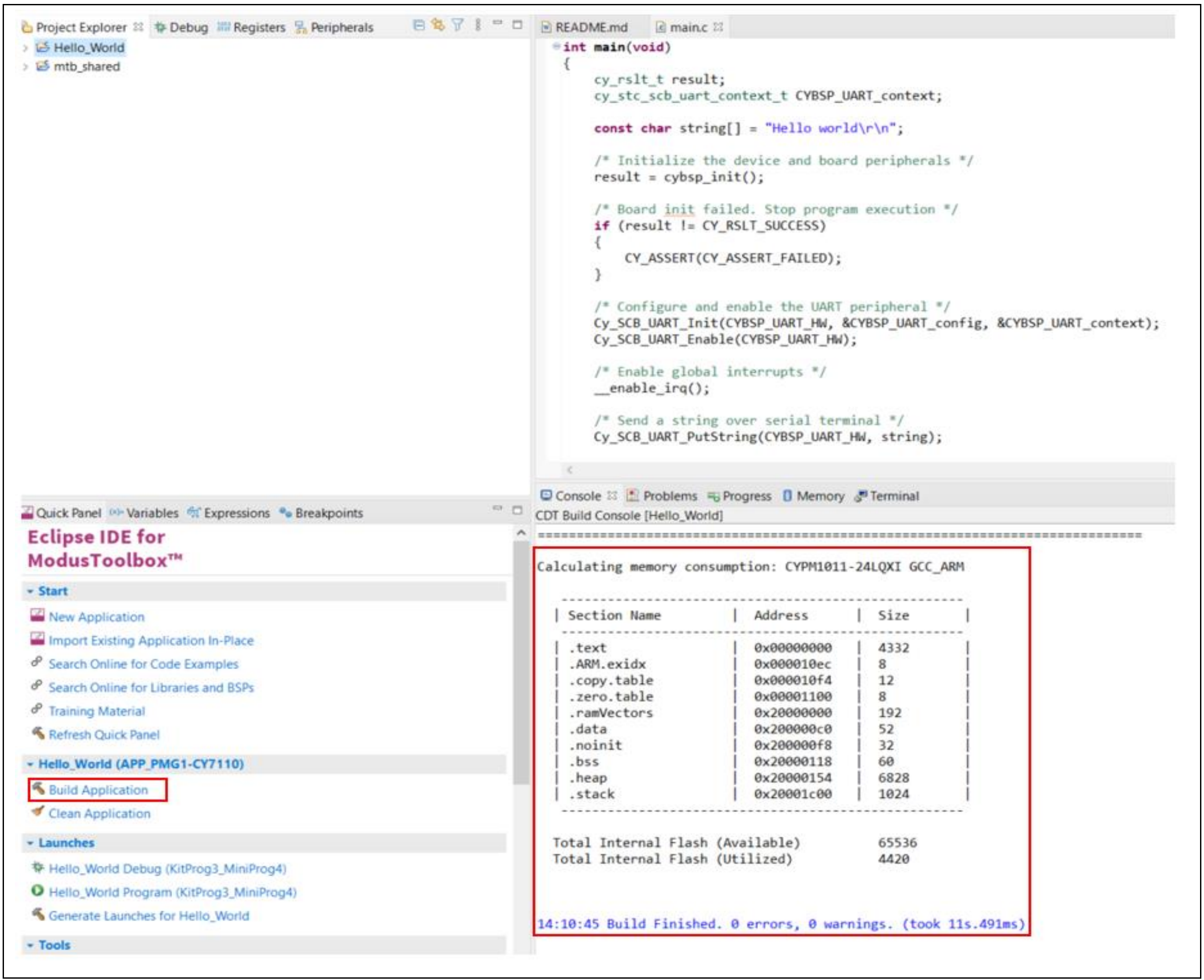


Figure 16 Build the application

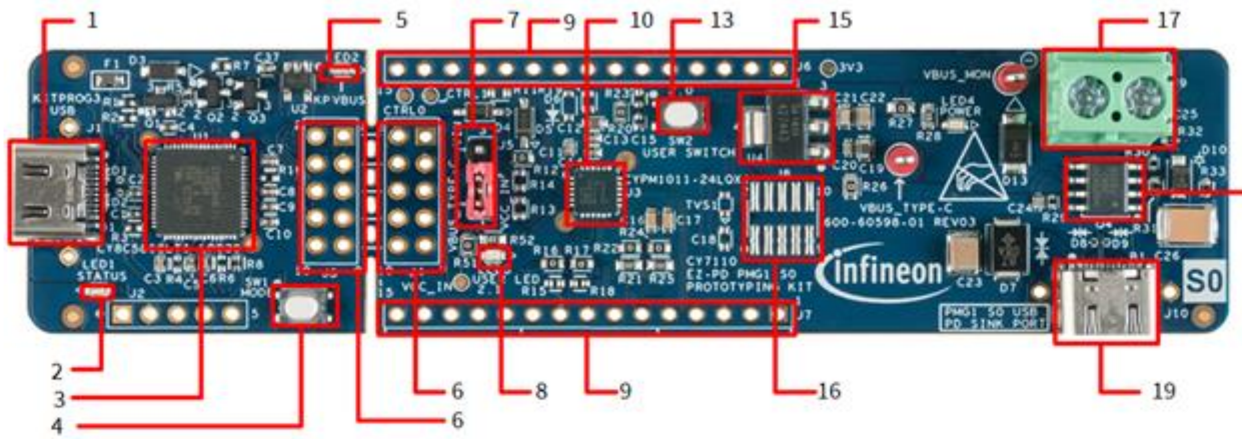
If you encounter errors, try the following options:

- Revisit earlier steps to ensure that you accomplished all the required tasks.
- Check the **Problems** tab to find more details about the errors and to solve them.
- From **Quick Panel**, click 'Clean' and try to build the project again.

Note: *You can also use the command line interface (CLI) to build the application. Please refer to ModusToolbox™* ***build system*** *section in the ModusToolbox™ user guide. This document can also be accessed in ModusToolbox™ software via* ***help menu > ModusToolbox™ General Documentation****.*

Programming the device.

Figure 17 for CY7110 EZ-PD™ PMG1-S0 Prototyping Kit. See the 'Kit operation' section in the PMG1 MCU web page.



1. KitProg3 USB Type-C port (J1)
2. KitProg3 status LED (LED1)
3. PSoC™ 5LP controller (U1)
4. KitProg3 mode switch (SW1)
5. KitProg3 VBUS LED (LED2)
6. KitProg3 header (J3, J4) *
7. Power selection jumper (J5)
8. User LED (LED3)
9. I/O headers (J6, J7) *

10. CYPM1011-24LQXI (PMG1-S0)
13. User switch (SW2)
15. 3.3V on-board LDO
16. 10-pin SWD/JTAG header *
17. DC_OUT terminal block (J9)
18. Load switch
19. PMG1 MCU USB Type-C port (J10)

* Footprint only; not populated on the board.

Figure 17 Choose CY7110 EZ-PD™ PMG1-S0 Prototyping Kit

3. ModusToolbox™ software uses the OpenOCD protocol to program and debug applications on PMG1 MCU devices.

If you are using a PMG1 MCU Prototyping Kit with a built-in programmer (KitProg3), follow steps 4 and 2.

If you are developing on your own hardware, you may need a hardware programmer/debugger; for example, an Infineon CY8CKIT-005 MiniProg4.

4. **Make the following hardware connections on the prototyping kit:**
   c) Connect a Type-C cable from the J1 (KitProg3 USB Type-C port) connector on the kit to the host.
   d) Connect the J5 (power selection jumper) pins 2 and 3 for programming mode.

Note: *Device can also be programmed by connecting J5 (power selection jumper) pins 1 and 2 for operation mode and connecting both Type-C cables J1 and J10 to the host and power supply respectively.*

5. **Program the board:**

a) In **ModusToolbox™ software**, select the application project in **Project Explorer**.

b) In **Quick Panel,** click **<Application Name> Program (KitProg3_MiniProg4)** as shown in Figure 18.

The IDE will select and run the appropriate run configuration. Note that this step will also perform a build if any files have been modified since the last build.

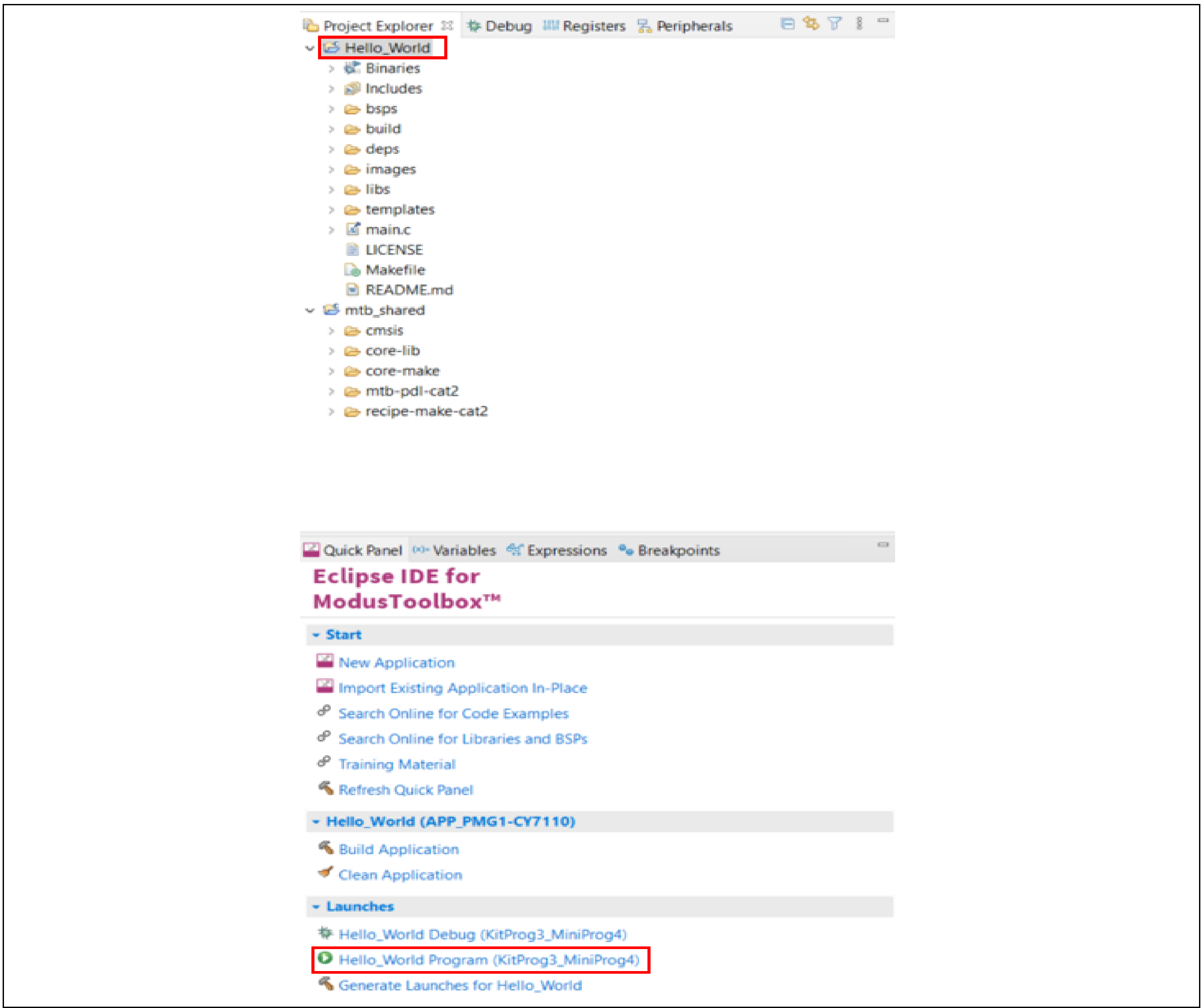


Figure 18 Programming an application to a device

6. The **Console** view lists the results of the programming operation, as Figure 19 shows.

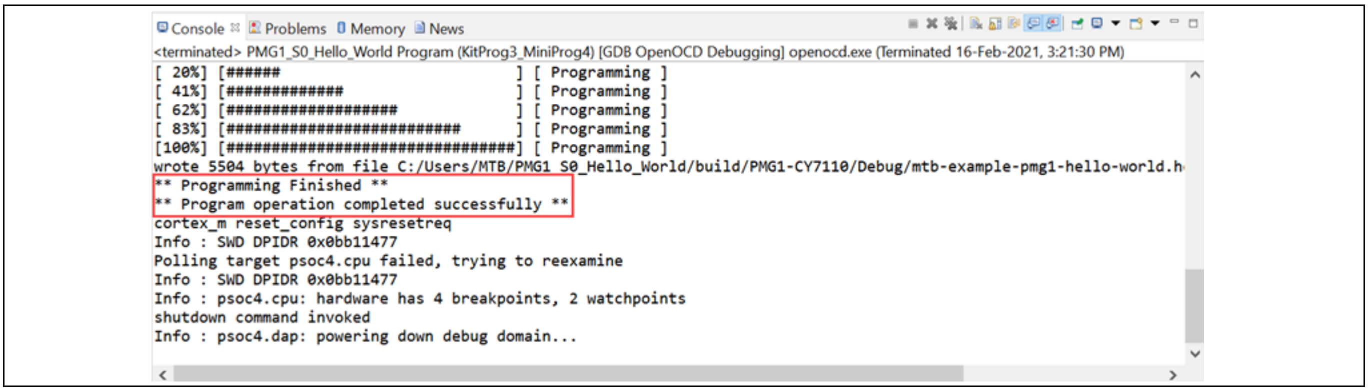


Figure 19 Console – programming results

3.7 Test your design

1. Make the following hardware connections:

a) Connect the J5 (Power Selection Jumper) pins 1 and 2 for operational mode.

b) Connect J6.10 to J3.8 and J6.9 to J3.10 to establish a UART connection between KitProg3 and PMG1-S0. Please follow the readme for UART connection with other kits

Note: *This step to establish UART connection needs to be executed only for the following older revision of the kit boards:*

- CY7110 board revision 3 or lower
- CY7111 board revision 2 or lower
- CY7112 board revision 2 or lower
- CY7113 board revision 3 or lower

The kit revision number (600-60xxx-01 Revxx) is marked on the silkscreen of the kit board as shown in Figure 20:

Figure 20 Identifying PMG1 MCU kit board revision number

c) Connect a Type-C cable from the J1 (KitProg3 USB Type-C port) connector on the kit to the host.

d) Confirm that KitProg3 VBUS LED (LED2) and status LED (LED1) glow in amber color.

2. Open a terminal program and select the KitProg3 COM port as shown in Figure 21.

   This application note uses Tera Term as the UART terminal emulator to view the results. You can use any terminal of your choice to view the output.

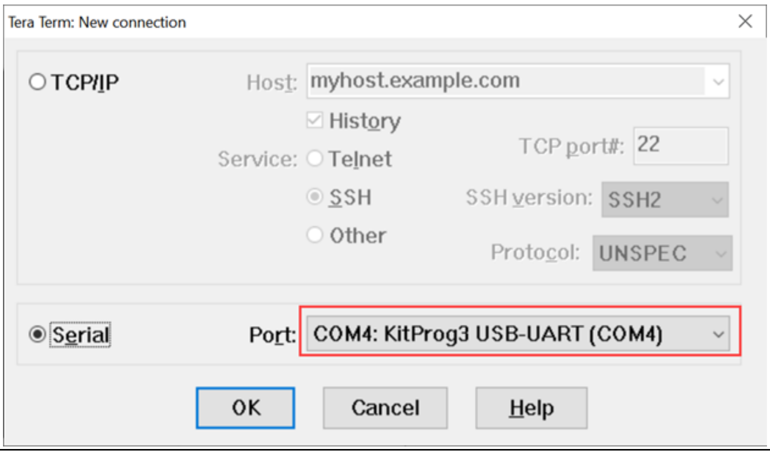


Figure 21 Selecting the KitProg3 COM port in Tera Term

3. **Set the serial port parameters to 8N1 and 115200 baud.**

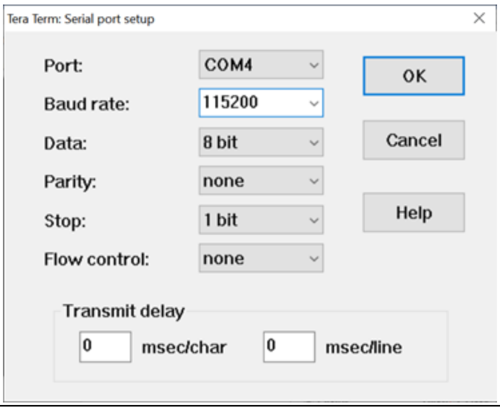


Figure 22 Configuring the baud rate in Tera Term

4. Connect a Type-C cable from the J10 (PMG1 MCU USB Type-C port) connector to the host.
5. Confirm that "Hello World" is displayed on the UART terminal.

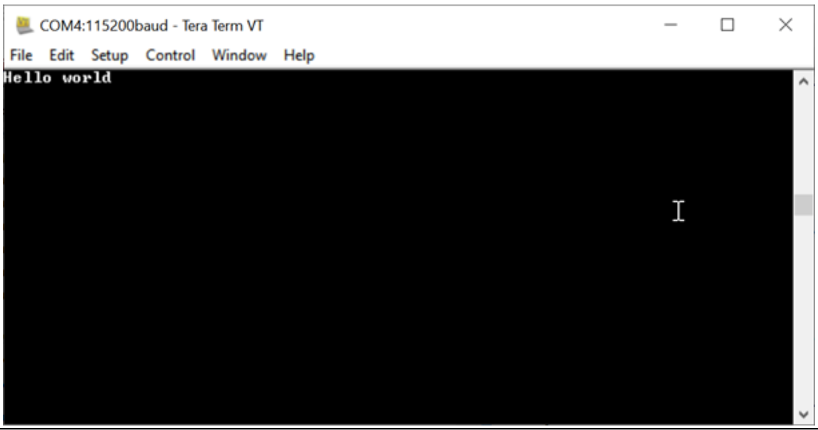


Figure 23 “Hello World” in terminal output

6. Confirm that the Power LED (LED4) glows in amber color and User LED (LED3) blinks.

3.8 Debug the application

Eclipse IDE for ModusToolbox™ software can be used to debug applications. The prototyping kits can be acquired in debug mode though either of the following three interfaces:

- KitProg3 USB Type-C port
- 10-pin SWD header
- I/O header pins

The kit needs to be configured for operational mode before starting the debug.

3.8.1 Debugging through the KitProg3 interface

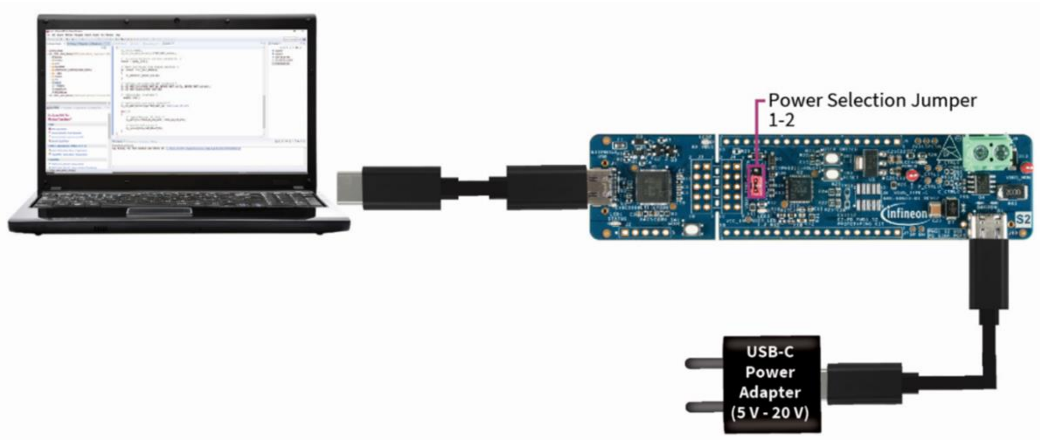


Figure 24 Debugging through KitProg3

Do the following to debug an application on theCY7110, CY7111, CY7112, and CY7113 Prototyping Kits over KitProg3 interface from Eclipse IDE for ModusToolbox™ software.

1. Place the jumper shunt on pins 1–2 of the power section jumper (J5) to configure the kit in operational mode.
2. Connect the USB-PD sink port to the USB PD source to activate on-board LDO, Load switch, and user LED. Ensure that LED4, which is power LED, glows green.
3. Connect the kit to the host PC through the KitProg3 USB Type-C port (J5).
4. Ensure that LED1 and LED2 glow in amber color.

   LED2, (KitProg3 Power LED) indicates that the KitProg3 module is powered.

   LED1, (status LED) indicates the programming/debug mode status and is ON when KitProg3 is powered.
5. Go to the **Quick Panel** tab. Click **<Application name> Debug (KitProg3_MiniProg4)** from the **Launches** section.

   The IDE will switch to debugging mode and will halt at the first line of the main () function. This indicates that the application is ready for debugging.

Alternatively,

1. Place the jumper shunt (J5) on pins 2-3 to enable ModusToolbox™ software to acquire the target (PMG1-S0) in power cycle mode.

2. Connect the **USB PD sink port** to the **USB PD source** to activate on-board LDO, Load switch, and user LED. Ensure that LED4, which is power LED, glows green.
3. Connect the kit to the host PC through programming KitProg3 USB Type-C port.
   Ensure that LED1 and LED2 glow in amber color.
   LED2, (KitProg3 Power LED), indicates that the KitProg3 module is powered.
   LED1, (status LED), indicates the programming/debug mode status and is ON when KitProg3 is powered.
4. Go to **Run** menu, click on **Debug Configurations**, as shown in Figure 25.
5. Under **GDB OpenOCD Debugging**, select <*Application name*> **Attach (KitProg3_MiniProg4)** and click **Debug** as shown in Figure 26.
   This will start a debugging session attaching to a running target without programming or reset. For more information on debug configurations refer ModusToolbox™ user guide.

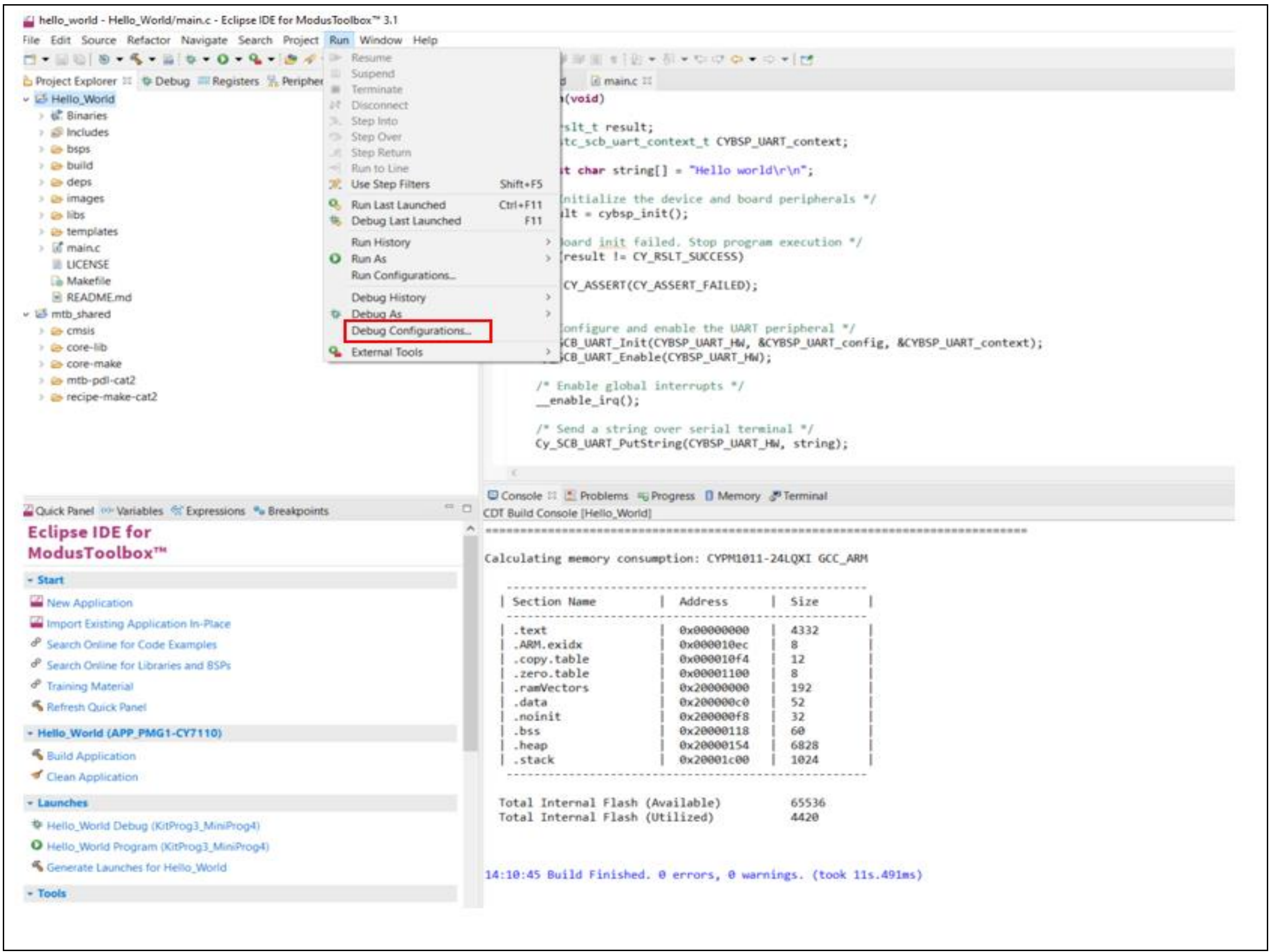


Figure 25 Selecting Debug configurations from the Run menu

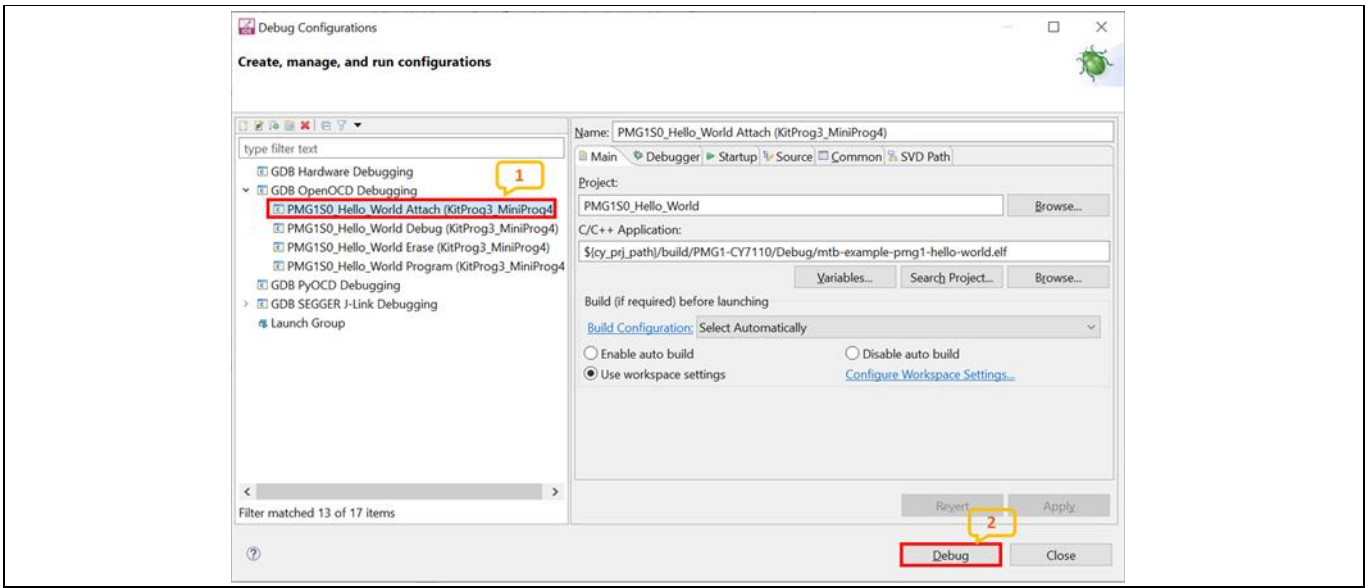


Figure 26 Selecting Debug Attach and starting a debug session

On successful completion of compilation, build, and debug launch, ModusToolbox™ software acquires the kit in debug mode.

3.8.2 Debug through the 10-pin SWD header

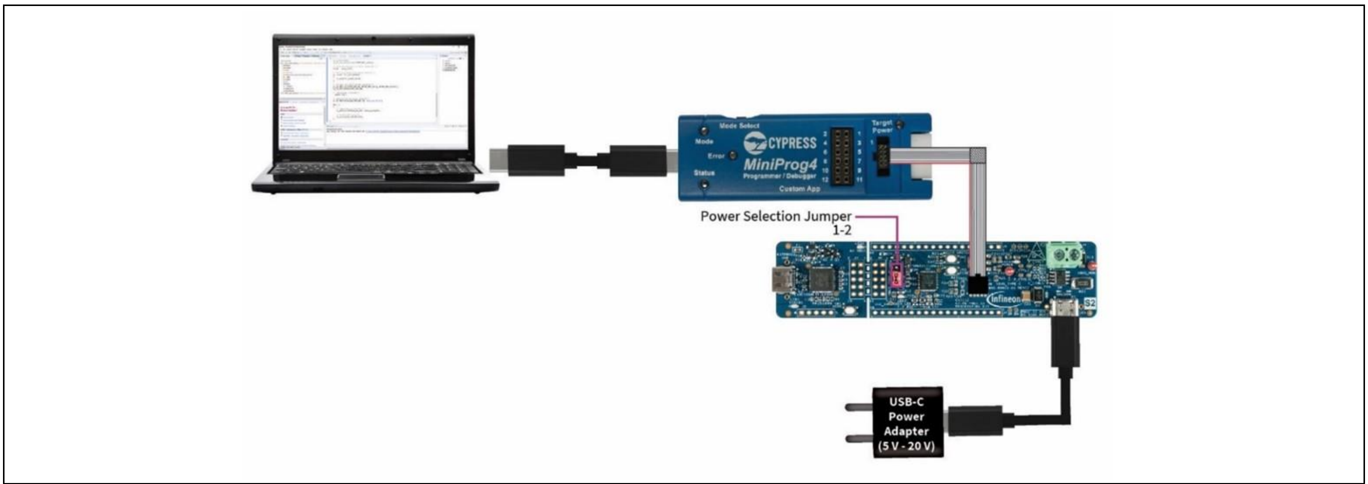


Figure 27 Debugging through the 10-pin SWD header

Figure 27 shows the setup for debugging prototyping kits through 10-pin SWD connector. The procedure to use Eclipse IDE for ModusToolbox™ software is the same as that of the KitProg3 interface. Follow steps 4 to 5 listed in Debugging through the KitProg3 interface.

3.9 Add middleware

The initialization of the middleware will be done in the *main.c* code.

1. In the **Quick Panel**, click on the **Library Manager** link.

2. In the subsequent dialog, select the **Add Library** tab.
3. Under **Core**, **Middleware, Peripheral and Utilities**, select and enable the required middleware.
4. Click **Ok** and **Update** as shown in Figure 28.

The files necessary to use the middleware are added in the *mtb_shared* folder.

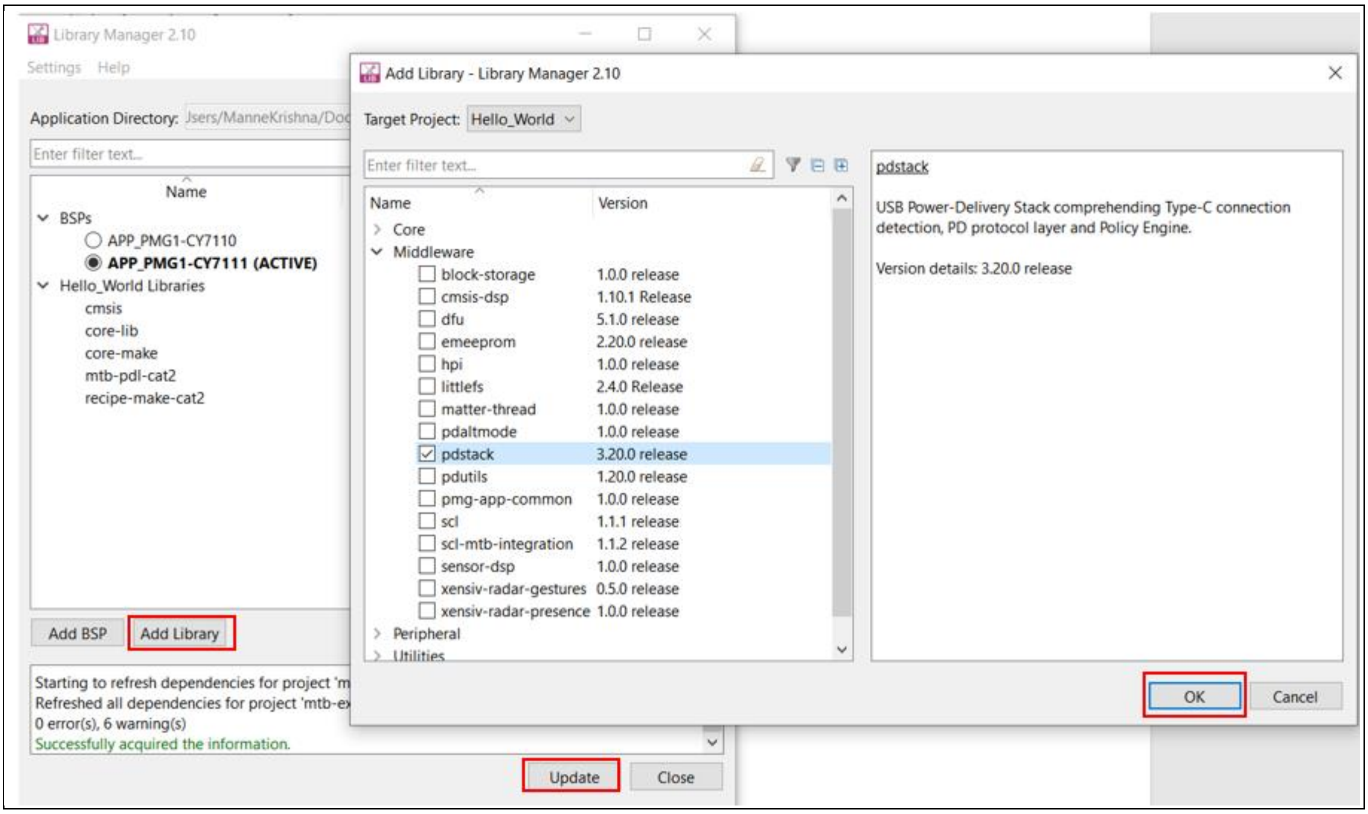


Figure 28 Add middleware

4 USB-PD sink example

A code example for USB-PD sink is available in the **Select New Application** option when the required BSP and "USBPD_Sink" application from the list is selected. See Figure 29.

The PMG1 MCU devices support a USBPD block which integrates the Type-C terminations, comparators, and Power Delivery Transceiver required to detect the attachment of a partner device and negotiate power contracts with them.

This application uses the PDStack in an UFP (Upstream Facing Port) - Sink configuration. The PMG1 MCU devices have a dead-battery Rd termination which ensures that a USB-C source/charger connected to it can detect the presence of a sink even when the PMG1 MCU device is not powered.

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

An overvoltage (OV) comparator in the USBPD block is used to detect cases where the power source is supplying incorrect voltage levels and automatically shut-down the power switches to protect the rest of the components on the board.

README.md file will be opened after the application is created successfully, which contains information about the template application implementation.

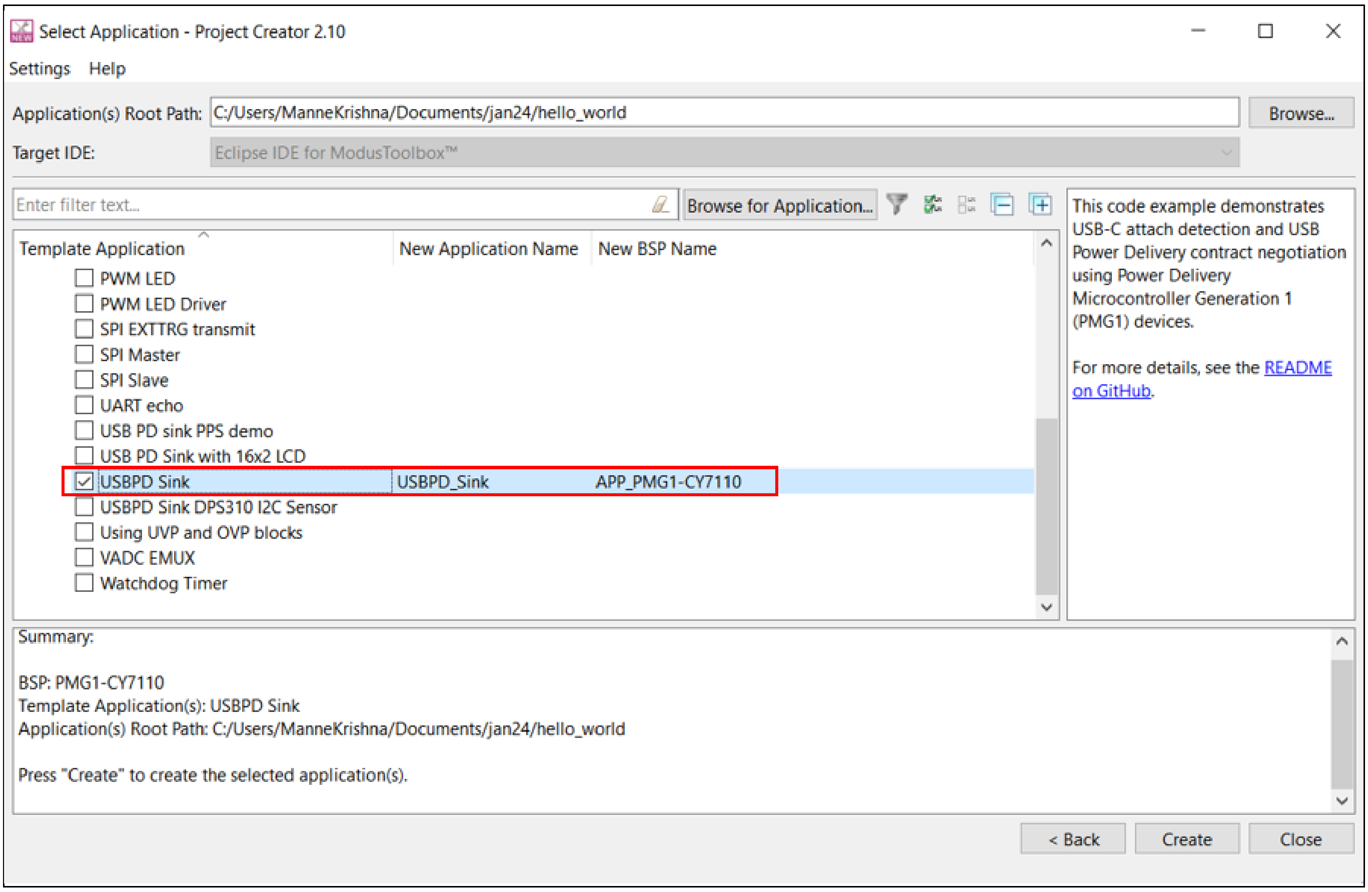


Figure 29 Selecting USBPD Sink CE

5 Porting the project across PMG1 MCUs

The project created for one MCU can be ported to work for other MCUs of the PMG1 MCU family. With a consideration that the feature/functionality/resource we are porting is available in the new MCU. Refer to the steps below for porting the project across PMG1 MCUs.

1. Open Library Manager from the Quick Link.
2. Do the following in Library Manager:
   a) Click on **Add BSP** to add new BSP and deselect the current one to remove it and select the BSP which is the active one. In Figure 30, shows that PMG1-CY7110 is deselected and **PMG1-CY7111** is selected. This means that you are porting the PMG1 MCU 'Hello World' code example to PMG1-S1 MCU.

Note: *To port the "Hello World" example to a custom hardware, choose the custom BSP that has been created.*

   b) Click **Update**.

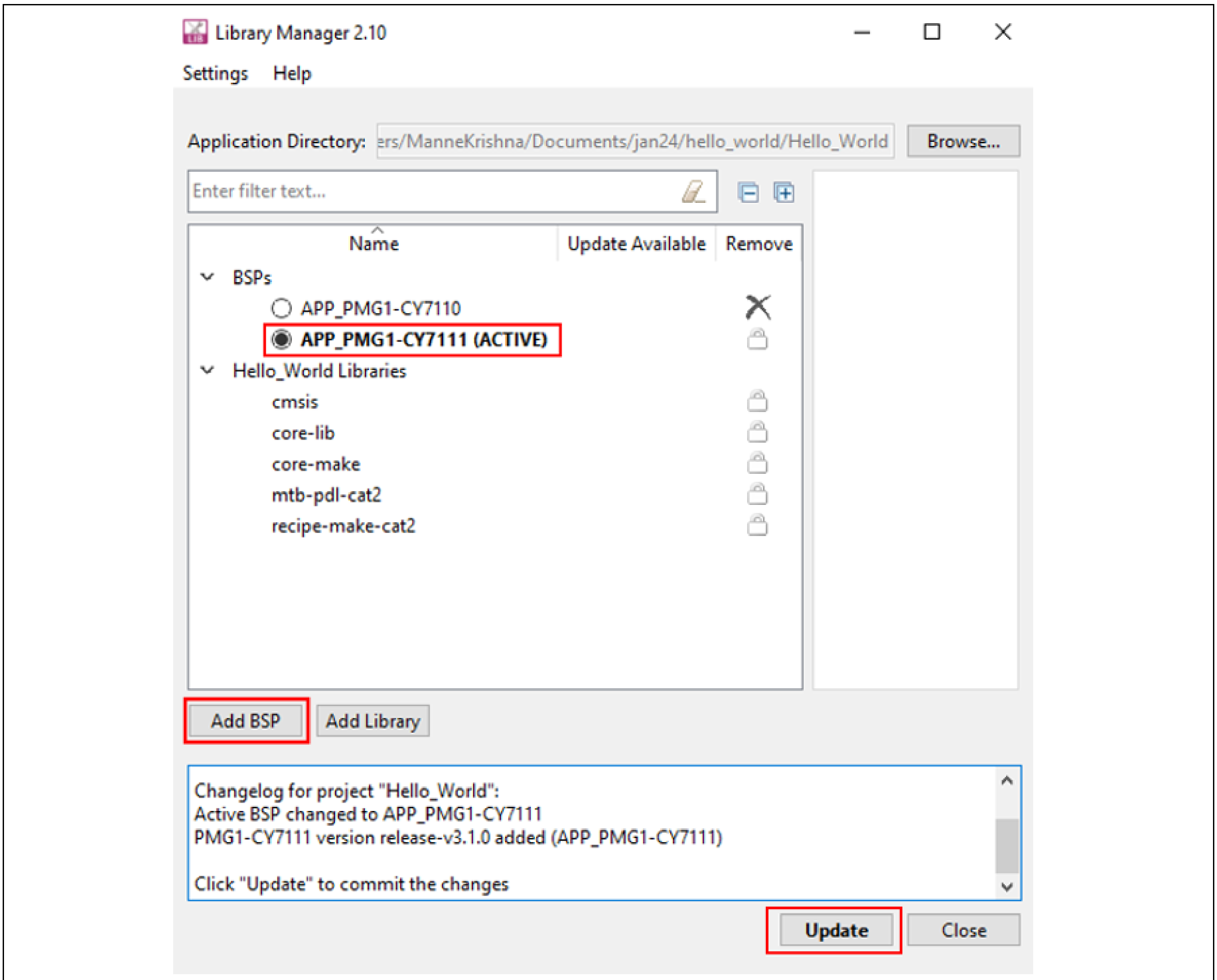


Figure 30 Porting a project to a different BSP

3. In the Makefile, the target name gets updated to the new BSP selected in previous step. See Figure 31.

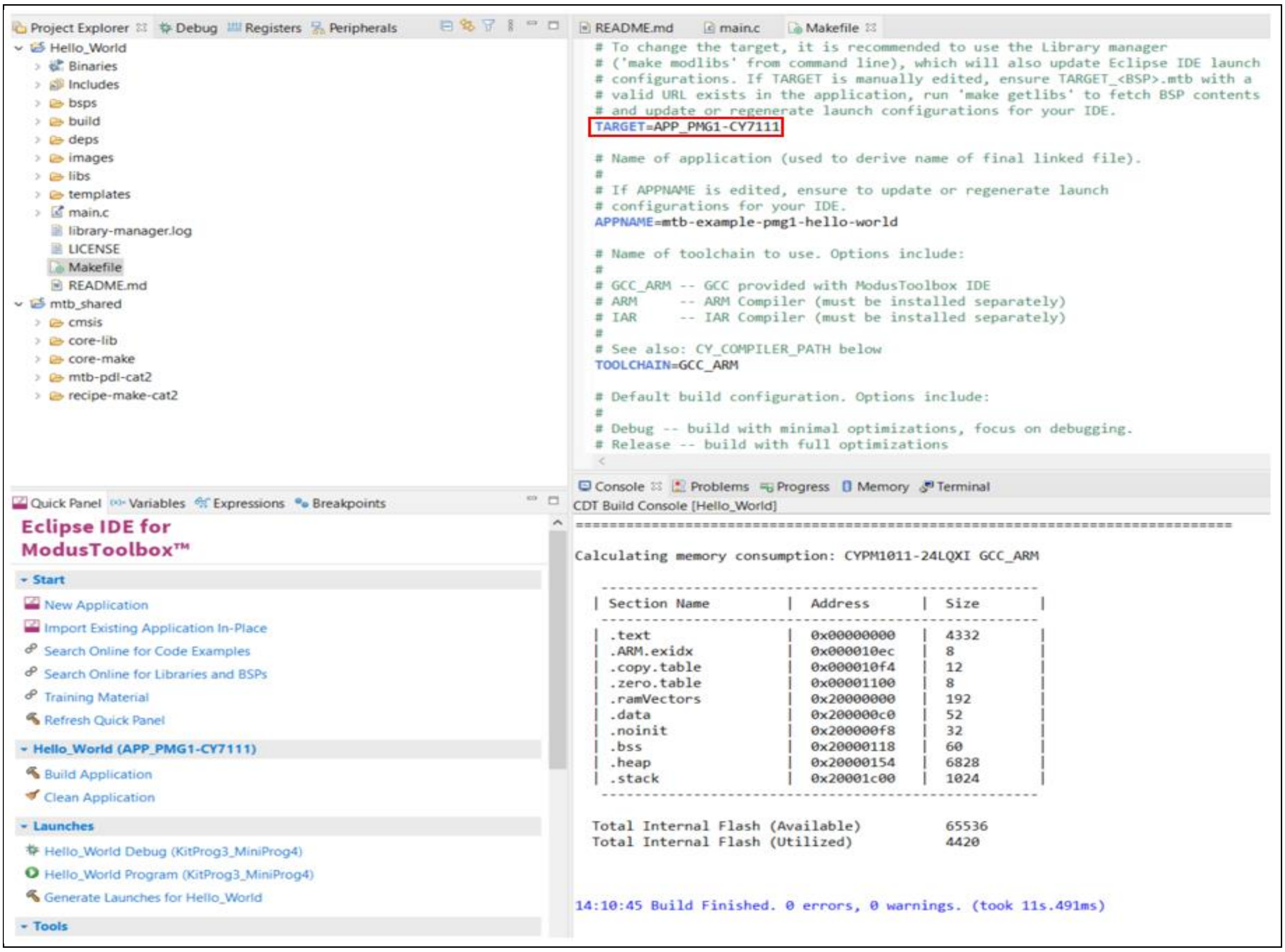


Figure 31 Edit the target name

4. Do the following in **Quick Panel**:
   a) Click **Generate Launches** for <project name>. See Figure 32.
   b) Click **Refresh Quick Panel**.

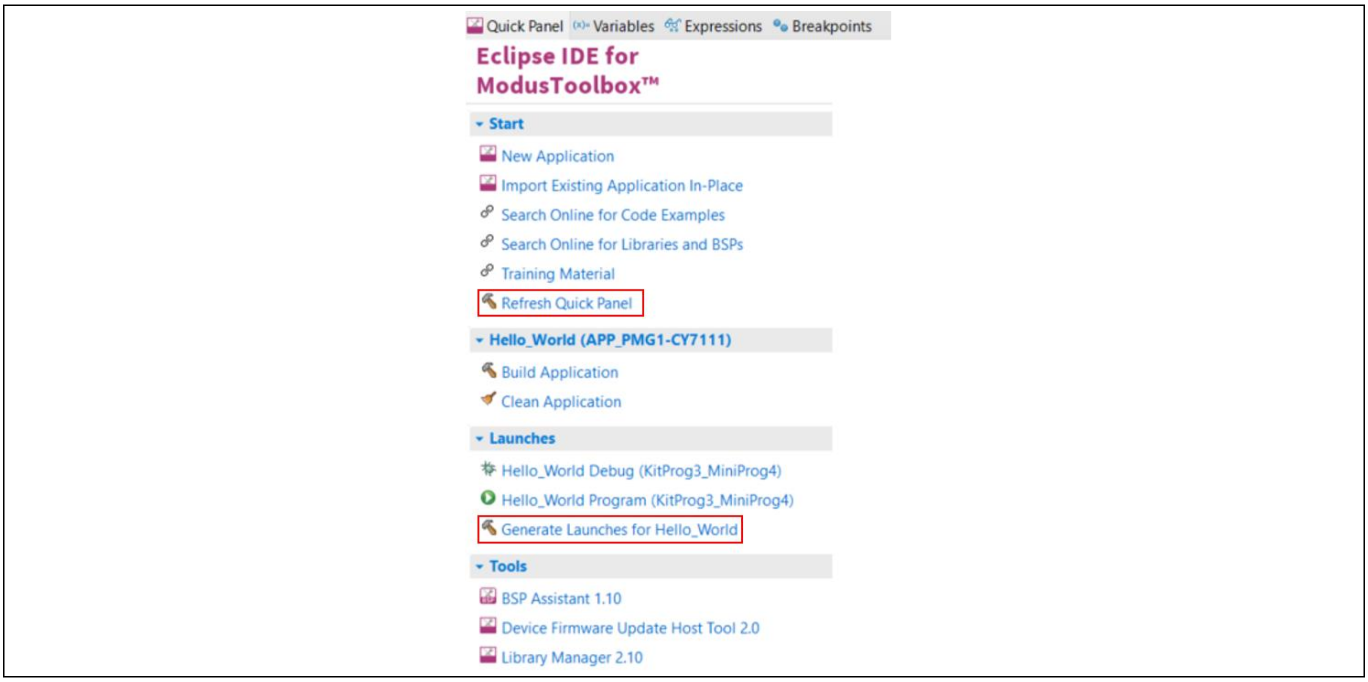


Figure 32 Regenerate launch configurations

Now, **Device Configurator** is updated to the new BSP, as shown in Figure 33.

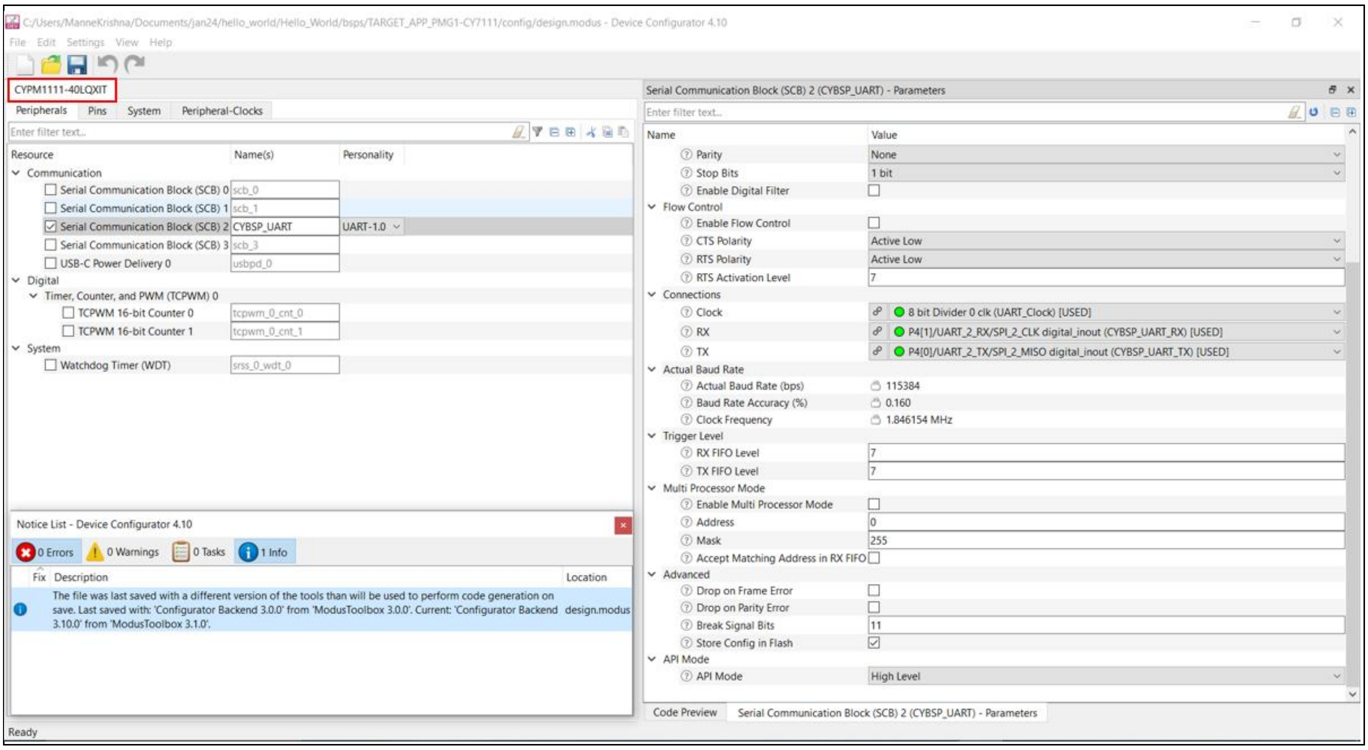


Figure 33 Device Configurator with new BSP

5. Continue with rest of the development steps from build, program, and to test the project.

6 Summary

This application note explores the capabilities of the EZ-PD™ PMG1 (Power Delivery Microcontroller Gen1) family of high-voltage MCU with USB-C PD and helps you to get started with your first project with PMG1 MCU in Eclipse IDE for ModusToolbox™ software.

Acronyms and abbreviations

Acronym	Expansion
AC	Apple Charging
AES	advanced encryption standard
AFC	adaptive fast charging
BC	battery charging
CDM	charged device model
CRC	cyclic redundancy check
HBM	human body model
OCP	overcurrent protection
OVP	overvoltage protection
OVT	overvoltage tolerant
PRNG	pseudo random number generation
RCP	reverse current protection. Supported in source configuration only.
SCP	short-circuit protection. Supported in source configuration only.
SHA	secure hash algorithm
TRNG	true random number generation
UVP	undervoltage protection

Glossary

This section lists the most commonly used terms that you might encounter while working with PMG1 MCU family of devices in ModusToolbox™ software.

Board support package (BSP): A BSP is the layer of firmware containing board-specific drivers and other functions. The board support package is a set of libraries that provide firmware APIs to initialize the board and provide access to board level peripherals.

KitProg3: The KitProg3 is an onboard programmer/debugger with USB-I2C and USB-UART bridge functionality. The KitProg3 is integrated onto most PMG1 MCU kits.

MiniProg3/**MiniProg4**: The Program and Debug Kit. This is helpful in programming or debugging the MCU, when you are developing your own hardware or KitProg3 is not working.

Personality: A personality expresses the configurability of a resource for a functionality. For example, the SCB resource can be configured to be an UART, SPI, or I2C personalities.

Middleware: Middleware is a set of firmware modules that provide specific capabilities to an application. This provides high level software interfaces to device features (e.g., PDStack).

ModusToolbox™ software: An Eclipse-based embedded design platform for developers that provides a single, coherent, and familiar design experience combining the driver libraries, middleware and consumes lowest power, and helps most flexible MCUs with best-in-class sensing.

Peripheral driver library: The peripheral driver library (PDL) simplifies software development for the PMG1 MCU architecture. The PDL reduces the need to understand register usage and bit structures, thus easing software development for the extensive set of peripherals available.

Revision history

Document revision	Date	Description of changes
**	2021-02-26	Initial release
*A	2021-07-28	Added 1.5 EZ-PD™ PMG1-S3 MCU section Updated Table 2 and Figure 12 Added a note in 3.6 Added 3.8 Debug the application section
*B	2022-02-10	Updated section 3.7 Test your design to describe the UART connection step on older revisions of the kit board
*C	2024-02-14	Updated section 2.2 Firmware/application development using ModusToolbox™ software Added section 2.7 Programming using mtb-programmer Updated the structure of the document Updated all block diagram to latest colour code Updated all the images to latest format

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

Edition 2024-02-14
Published by

Infineon Technologies AG
81726 Munich, Germany

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

Document reference
002-32553 Rev. *C