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

About this document

Scope and purpose

This application note introduces you to 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 anyone who uses the EZ-PD™ PMG1 MCU family on ModusToolbox™ software.

Table of contents

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 power delivery (PD) microcontrollers (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 your application.

Table 1 Comparison of features of different MCUs of 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 in 48QFN 2 ports in 97BGA
	Role	Sink	DRP	DRP	DRP
	MOSFET gate drivers	1x PFET	2x PFET	2x NFET	Flexible 2x NFET
	Fault protection	VBUS OVP and UVP	VBUS OVP, UVP and OCP. SCP and RCP (Source only)	VBUS OVP, UVP and OCP	VBUS OVP, UVP and OCP. SCP and RCP (Source only)
USB	Integrated Full Speed USB 2.0 device with Billboard Class support	No	No	Yes	Yes
Voltage range	Supply (V)	VDDD (2.7–5.5) VBUS (4–21.5)	VSYS (2.75–5.5) VBUS (4–21.5)	VSYS (2.7–5.5) VBUS (4–21.5)	VSYS (2.8–5.5) VBUS (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 in 48QFN (out of which only 5 can be configured as SPI and UART) 8 in 97BGA
	TCPWM block (configurable as timer, counter or	4	2	4	7 in 48QFN 8 in 97BGA

Subsystem or range	Item	PMG1-S0	PMG1-S1	PMG1-S2	PMG1-S3
	pulse width modulator)				
	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 OVT)	17 (15+2 OVT)	20 (18+2 OVT)	26 (24+2 OVT) in 48QFN 50 (48+2 OVT) in 97BGA
Charging standards	Charging source	-	BC 1.2, AC	BC 1.2, AC	BC 1.2, AC, AFC and Quick Charge 3.0
	Charging sink	BC 1.2, Apple Charging (AC)	BC 1.2, AC	BC 1.2, AC	BC 1.2, AC
ESD protection	ESD protection	Yes (Up to ± 8-kV contact discharge and up to ±15-kV air discharge, HBM, CDM)	Yes (HBM and CDM)	Yes (Up to ± 8-kV contact discharge and up to ±15-kV air discharge, HBM, CDM)	Yes (HBM and CDM)
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)	48QFN (6x6 mm, 0.5-mm pitch) 97BGA (6x6 mm, 0.5-mm and 0.65-mm pitch)

1.2 PMG1-S0 MCU

PMG1-S0 MCU is the first member of the PMG1 MCU family. It includes a 48-MHz Arm® Cortex®-M0 processor with 64-KB flash, a complete Type-C USB-PD transceiver, pull-down termination resistor R_D to support Sink on a Type-C port, 12 GPIOs, and system-level ESD protection. It is available in a 24-pin QFN package.

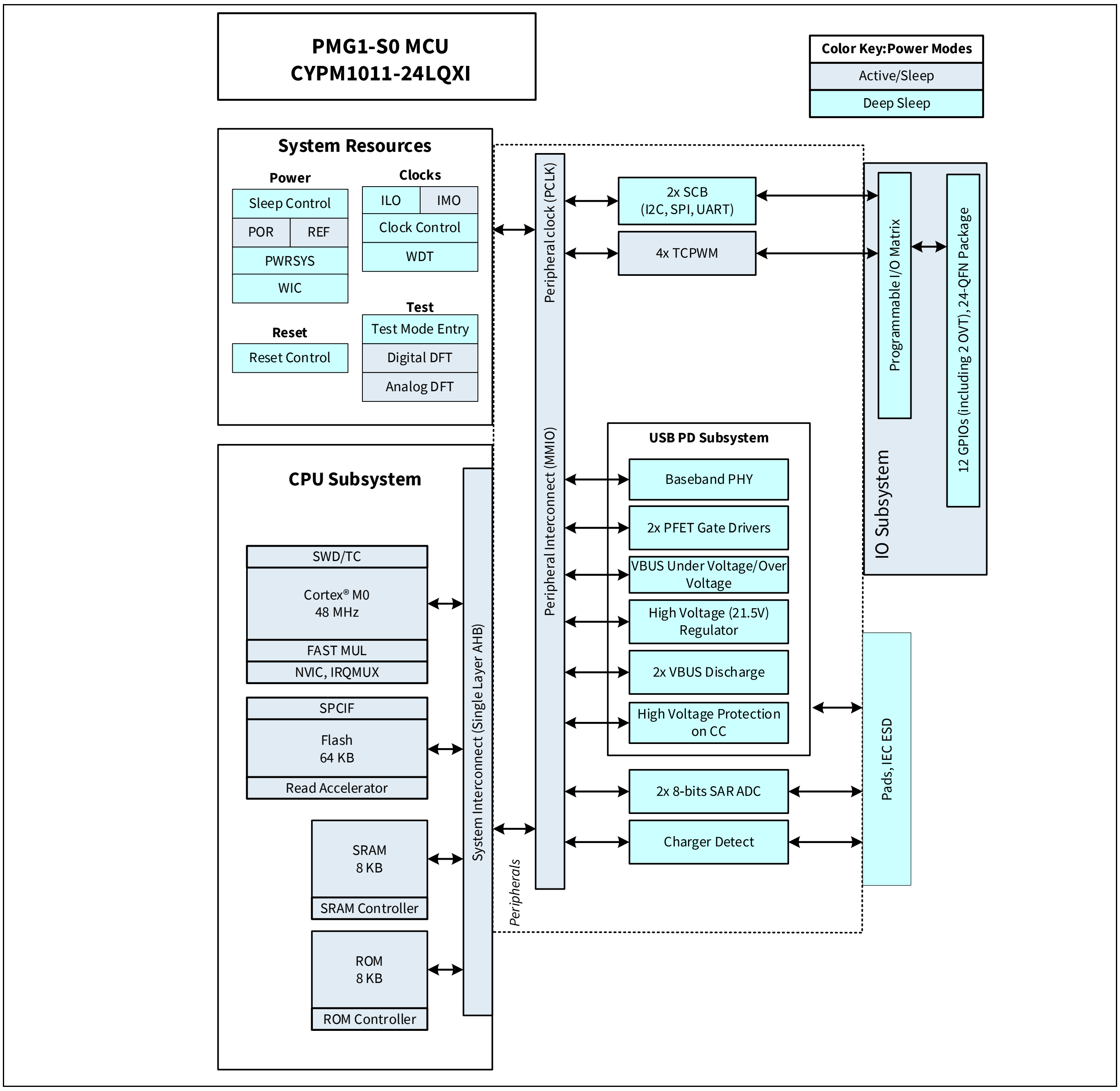


Figure 1 **PMG1-S0 MCU block diagram**

1.3 PMG1-S1 MCU

PMG1-S1 MCU includes a 48-MHz Arm® Cortex®-M0 processor but with a higher flash size than PMG1-S0 MCU. It has 128-KB flash, a complete Type-C USB PD transceiver with all termination resistors R_P, R_D, and dead battery R_D, and 17 GPIOs. It is available in a 40-pin QFN package.

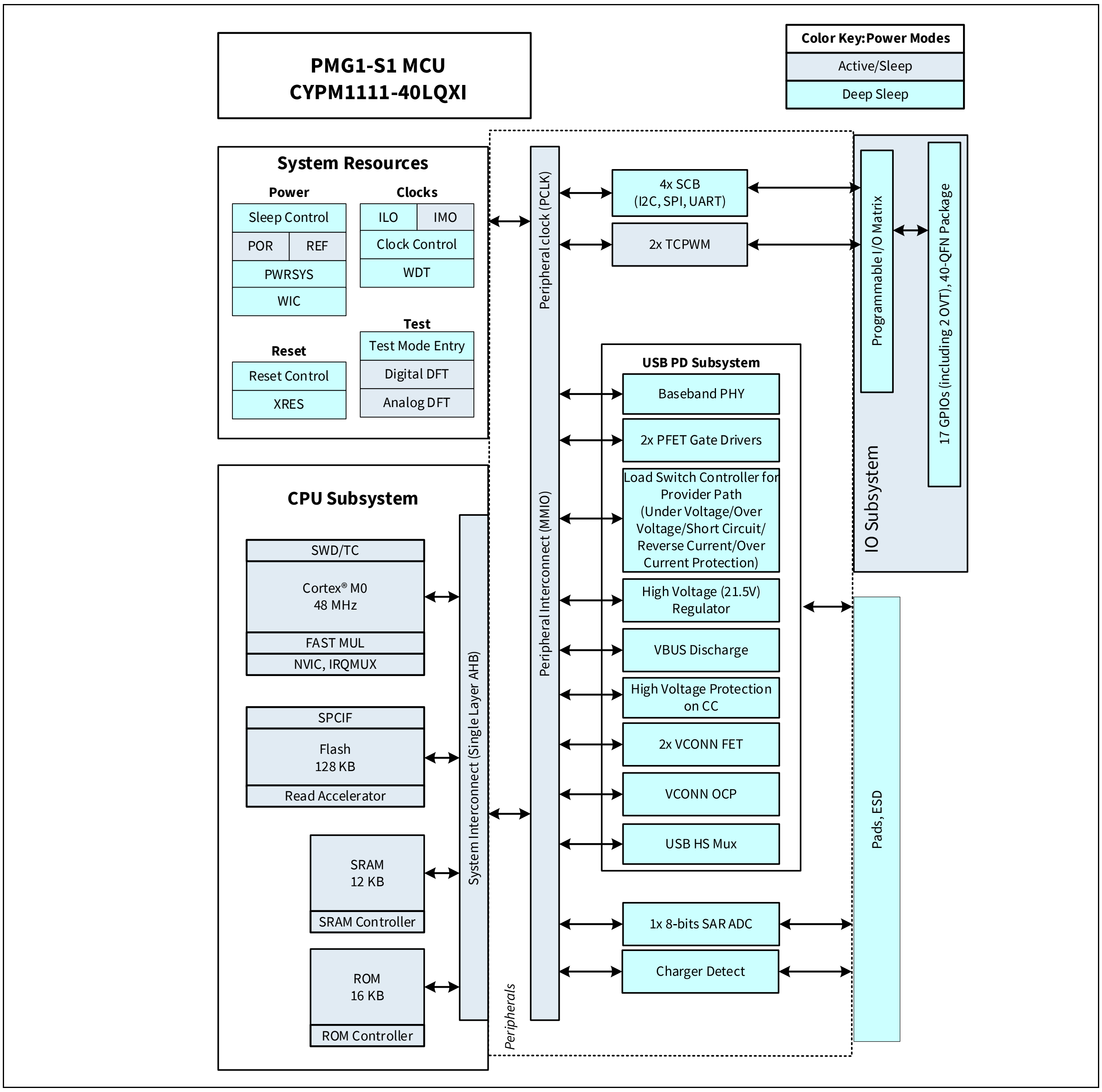


Figure 2 **PMG1-S1 MCU block diagram**

1.4 PMG1-S2 MCU

PMG1-S2 MCU uses the 48-MHz Arm® Cortex®-M0 processor with 128-KB flash and includes a Full Speed USB device controller, a Crypto engine for authentication, 20 GPIOs and system level ESD protection. PMG1-S2 MCU is available in 40-QFN package.

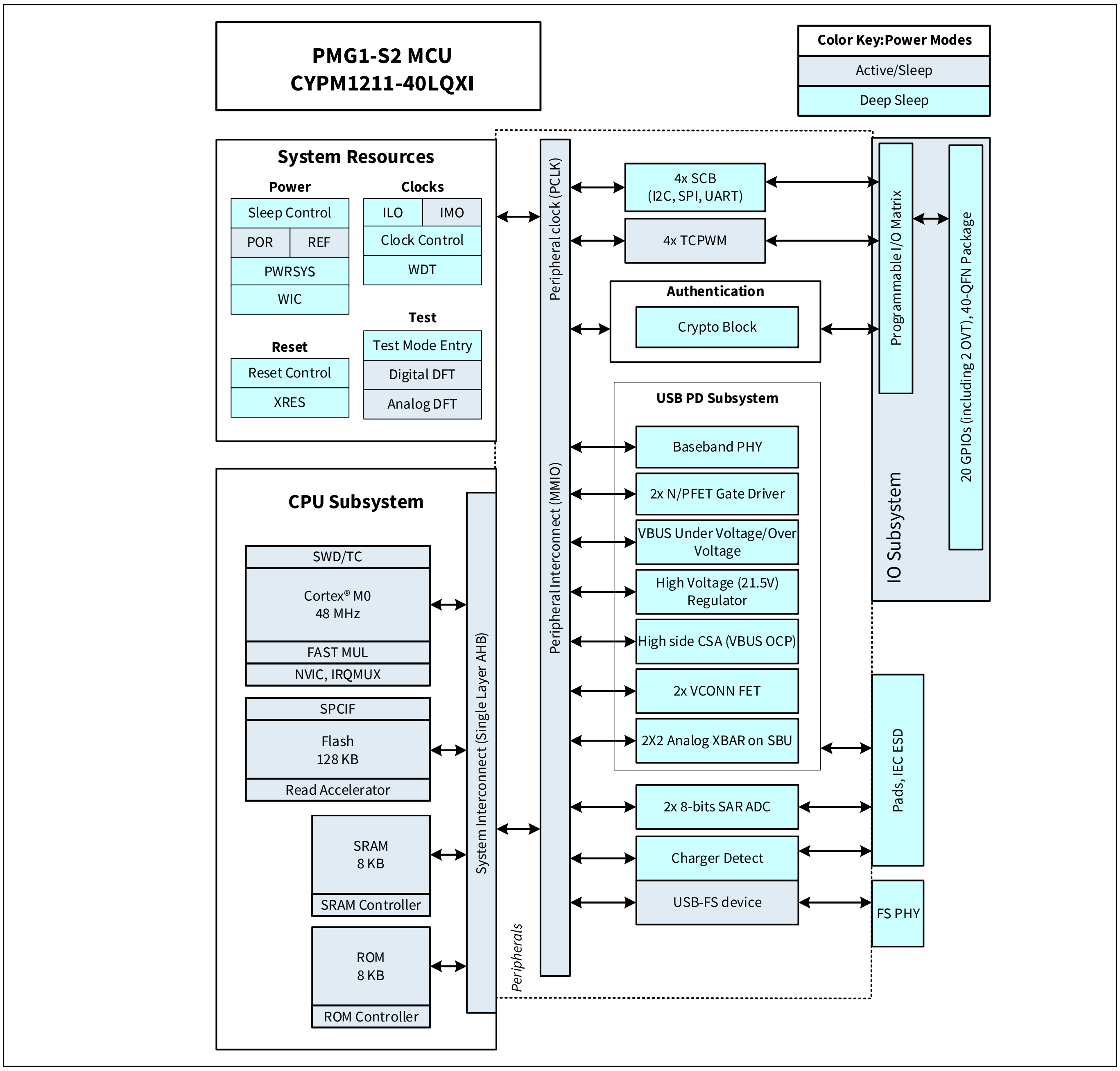


Figure 3 PMG1-S2 MCU block diagram

1.5 PMG1-S3 MCU

PMG1-S3 MCU uses the 48-MHz Arm® Cortex®-M0+ processor with 256-KB flash and includes a Full Speed USB device controller, CAPSENSE™, DMA, a Crypto engine for authentication, 26/50 GPIOs, and system-level ESD protection. PMG1-S3 MCU is available in 48-QFN and 97-BGA package.

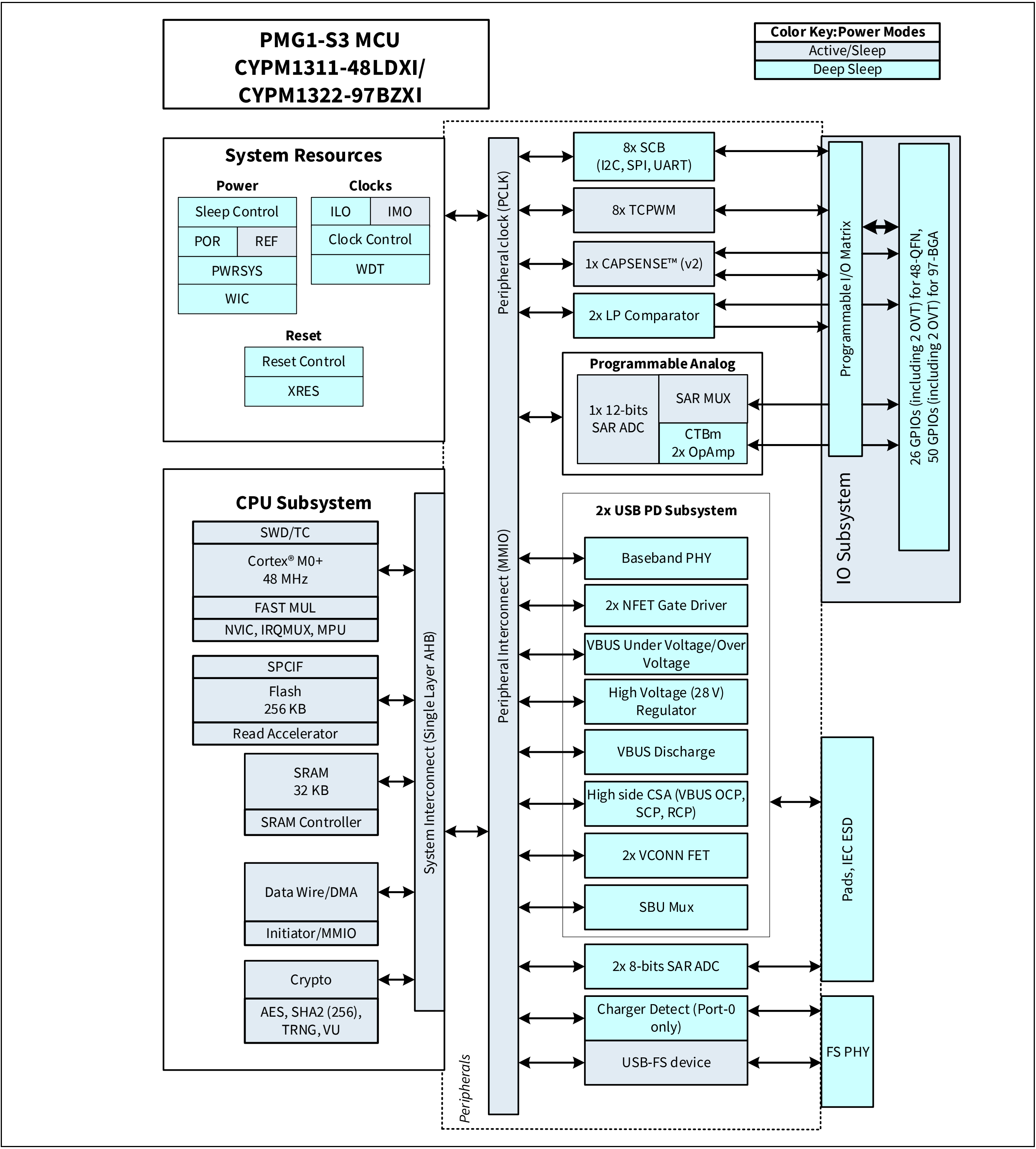


Figure 4 **PMG1-S3 MCU block diagram**

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 world-class 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 will be 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. The following is an abbreviated list of resources for PMG1 MCU. Visit **PMG1 MCU webpage** to find out more.

- PMG1 MCU webpage
- Datasheets provide all the information needed to select and use a particular device, including functional description and electrical specifications.
- Application notes and code examples cover a broad range of topics, from basic to advanced level.
- Technical reference manuals (TRMs) 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 Firmware/application development using ModusToolbox™ software

ModusToolbox™ development platform 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, low-level drivers, 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 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™ software user guide**.

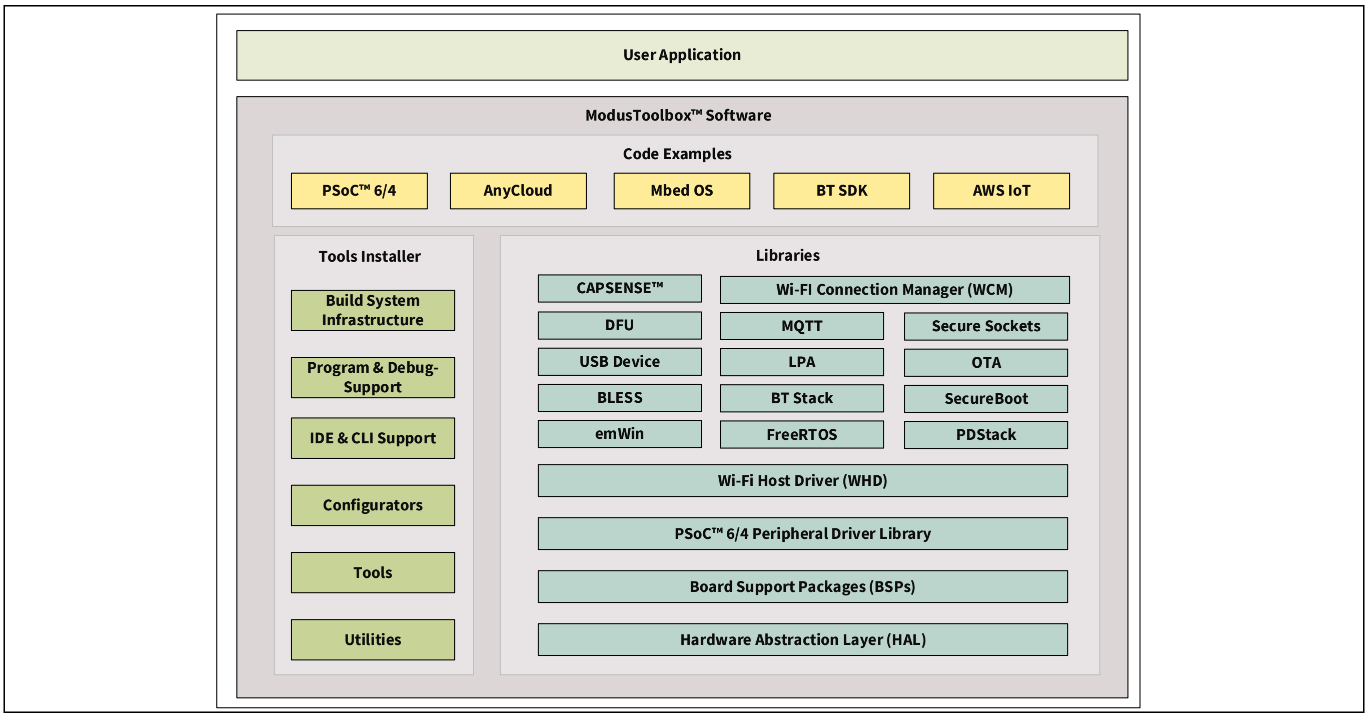


Figure 5 **ModusToolbox™ software**

Embedded PMG1 MCU is the native development flow supported by the Eclipse IDE for ModusToolbox™ software for PMG1 MCU. You can use it on the bare metal or with an RTOS of your choosing.

Following are the low-level resources for PMG1 MCU.

1. **Board Support Packages (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 APIs to initialize the board and provide access to board-level peripherals. It includes low-level resources such as Peripheral Driver Library (PDL) for PMG1 MCU and has macros for board peripherals. Custom BSPs can be created to enable support for end-application boards. See the **Board support Packages** section in **ModusToolbox™ software user guide** for more information.
2. **Peripheral Driver Library (PDL):** The PDL integrates device header files, start-up code, and peripheral drivers into a single package. The drivers abstract the hardware functions into a set of easy-to-use APIs. This reduces the need to understand register usage and bit structures, thus easing software development for the extensive set of peripherals in the PMG1 MCU series. You configure the driver for your application, and then use API calls to initialize and use the peripheral. For a PMG1 MCU solution, the on-chip blocks including USBPD and SCBs are controlled by the MTB CAT2 PDL .
3. **Middleware:** The middleware is a set of firmware modules that provide specific capabilities to an application. PDStack is the middleware available for PMG1 MCU, which contains the Type-C and USB-PD state machine implementations, the PD protocol, and PD Policy Manager blocks.

All low-level resources are delivered as libraries via GitHub repositories.

Figure 6 shows the blocks of ModusToolbox™ software which are relevant for PMG1 MCU.

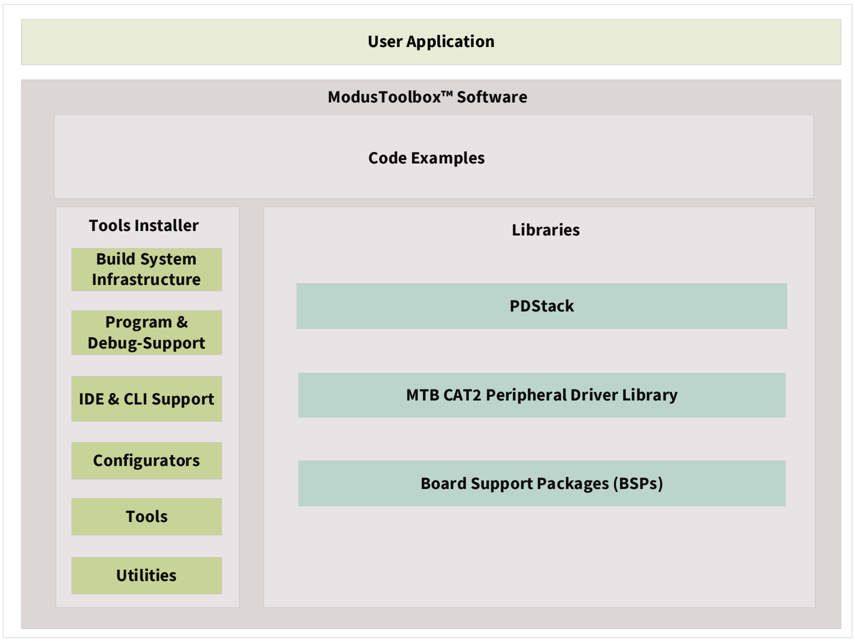


Figure 6 ModusToolbox™ software for PMG1 MCU

2.2.1 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.

2.2.1.1 Configurators

ModusToolbox™ software provides graphical applications 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, open the 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 standalone, in conjunction with other tools, or within a complete IDE. Configurators are used for:

- Setting options and generating code to configure 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 include the following:

- **Device Configurator:** Set up the system (platform) functions, as well as the basic peripherals (e.g., UART, Timer, PWM).
- **EZ-PD™ Configurator:** Provides a user-friendly tool for selecting the features of PDStack middleware and configuring parameters to generate the required code.

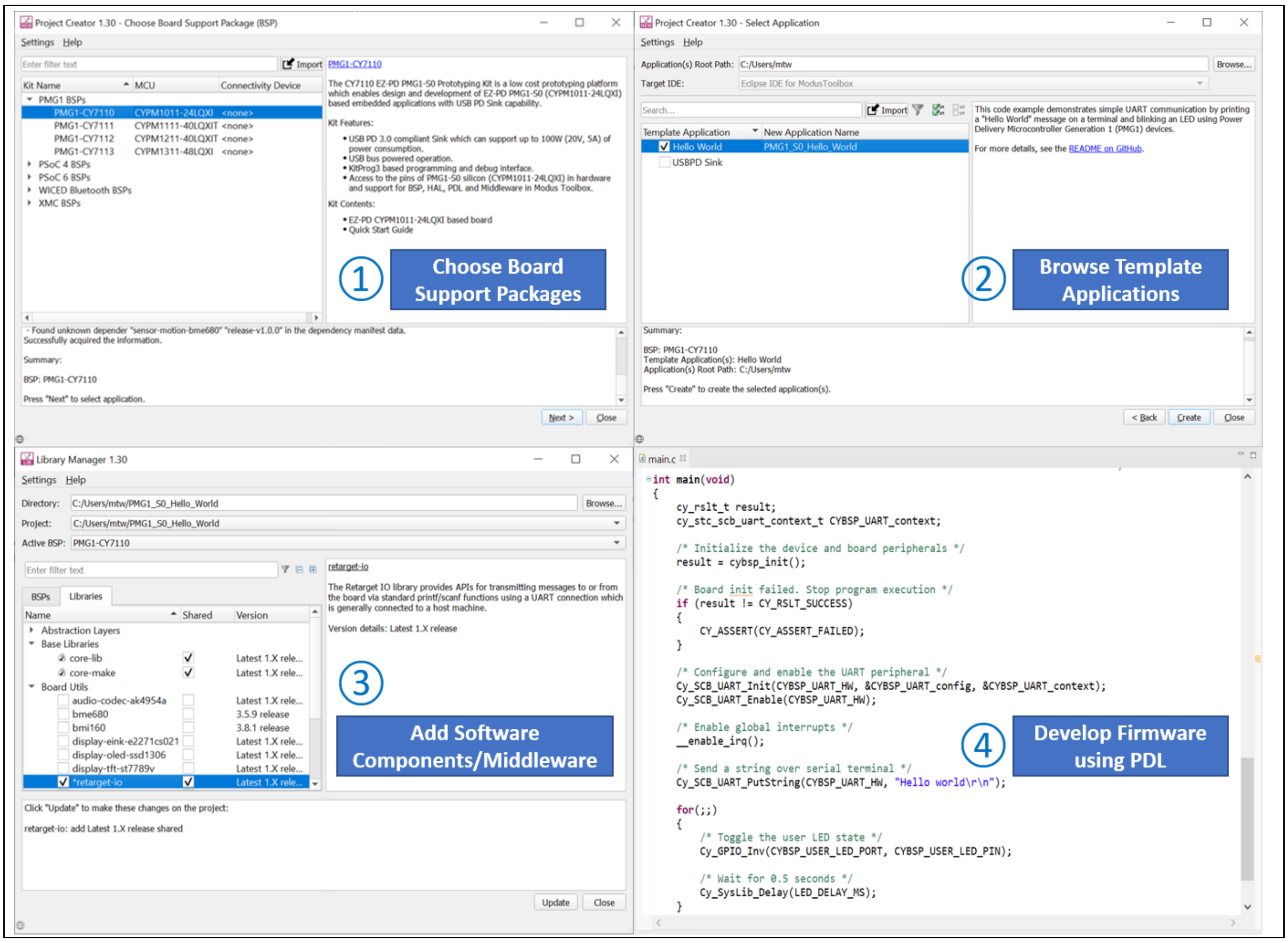
These configurators create their own files (e.g., *design.mtbezpd* for EZ-PD™ Configurator). The configurator file (*design.modus*) usually provided with the BSP and *design.mtbezpd* is provided as part of the code example. When an application is created based on a BSP, the files are copied into the application. You can also create custom device configurator files for an application and override the ones provided by the BSP. For more details refer to Documentation section in the **ModusToolbox™ tool** page. These documents can also be accessed in ModusToolbox™ **via help menu > ModusToolbox™ General Documentation > ModusToolbox™ Documentation Index.**

2.2.1.2 PMG1 MCU software development

Infineon provides customers with source code and tools to enable software development for PMG1 MCUs. You use tools to specify how you want to configure the hardware, generate code for that purpose which you use in your firmware, and include various middleware libraries for additional functionality, like PDStack. This source code makes it easier to develop the firmware for supported devices. It helps you quickly customize and build firmware without the need to understand the register set.

As **Figure 7** shows, with the Eclipse IDE for ModusToolbox™ software, you can:

1. Choose a board support package (BSP).
2. Create a new application based on a list of template applications, filtered by kit.
3. Add middleware.
4. Develop your application firmware using the PDL.


Project Creator 1.30 - Choose Board Support Package (BSP)
Settings Help
Enter filter text
Import
1 Choose Board Support Packages
Next >
Close
Project Creator 1.30 - Select Application
Settings Help
Search...
Import
Browse...
2 Browse Template Applications
< Back
Create
Close
Library Manager 1.30
Settings Help
Enter filter text
BSPs
Libraries
3 Add Software Components/Middleware
Update
Close
main.c
4 Develop Firmware using PDL

Kit Name	MCU	Connectivity Device
PMG1 BSPs		
PMG1-CY7110	CYPM1011-24LQXI	<none>
PMG1-CY7111	CYPM1111-40LQXIT	<none>
PMG1-CY7112	CYPM1211-40LQXIT	<none>
PMG1-CY7113	CYPM1311-48LQXI	<none>
PSoC 4 BSPs		
PSoC 6 BSPs		
WICED Bluetooth BSPs		
XMC BSPs		

PMG1-CY7110

The CY7110 EZ-PD PMG1-S0 Prototyping Kit is a low cost prototyping platform which enables design and development of EZ-PD PMG1-S0 (CYPM1011-24LQXI) based embedded applications with USB PD Sink capability.

Kit Features:

- USB PD 3.0 compliant Sink which can support up to 100W (20V, 5A) of power consumption.
- USB bus powered operation.
- KitProg3 based programming and debug interface.
- Access to the pins of PMG1-S0 silicon (CYPM1011-24LQXI) in hardware and support for BSP, HAL, PDL and Middleware in Modus Toolbox.

Kit Contents:

- EZ-PD CYPM1011-24LQXI based board
- Quick Start Guide

- Found unknown depender "sensor-motion-bme680" "release-v1.0.0" in the dependency manifest data.
Successfully acquired the information.

Summary:

BSP: PMG1-CY7110

Press "Next" to select application.

Application(s) Root Path: C:/Users/mtw

Target IDE: Eclipse IDE for ModusToolbox

Template Application	New Application Name
✓ Hello World	PMG1_S0_Hello_World
USBPD Sink	

This code example demonstrates simple UART communication by printing a "Hello World" message on a terminal and blinking an LED using Power Delivery Microcontroller Generation 1 (PMG1) devices.

For more details, see the README on GitHub.

Summary:

BSP: PMG1-CY7110
Template Application(s): Hello World
Application(s) Root Path: C:/Users/mtw

Press "Create" to create the selected application(s).

Directory: C:/Users/mtw/PMG1_S0_Hello_World

Project: C:/Users/mtw/PMG1_S0_Hello_World

Active BSP: PMG1-CY7110

Name	Shared	Version
Abstraction Layers		
Base Libraries		
core-lib	✓	Latest 1.X rele...
core-make	✓	Latest 1.X rele...
Board Utils		
audio-codec-ak4954a		Latest 1.X rele...
bme680		3.5.9 release
bmi160		3.8.1 release
display-eink-e2271cs021		Latest 1.X rele...
display-oled-ssd1306		Latest 1.X rele...
display-tft-st7789v		Latest 1.X rele...
✓ *retarget-io	✓	Latest 1.X rele...

retarget-io

The Retarget IO library provides APIs for transmitting messages to or from the board via standard printf/scanf functions using a UART connection which is generally connected to a host machine.

Version details: Latest 1.X release

Click "Update" to make these changes on the project:

retarget-io: add Latest 1.X release shared

```c
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_USER_LED_PORT, CYBSP_USER_LED_PIN);

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

Figure 7 Eclipse IDE for ModusToolbox™ software resources and middleware

2.3 ModusToolbox™ software 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 > ModusToolbox™ General Documentation or Help > Eclipse IDE for ModusToolbox™ Documentation**

2.4 Support for other IDEs

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

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

2.5 Programming and debugging

The Eclipse IDE of ModusToolbox™ software supports 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**.

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 Support" section in the **ModusToolbox™ software user guide**.

2.6 PMG1 MCU kits

Table 2 lists the list of PMG1 MCU kits and the corresponding BSPs.

Table 2 BSPs for PMG1 MCU kits

PMG1 MCU	PMG1 MCU Kit	BSP
CYPM1011-24LQXI	CY7110 EZ-PD™ PMG1-S0 prototyping kit	PMG1-CY7110
CYPM1111-40LQXI	CY7111 EZ-PD™ PMG1-S1 prototyping kit	PMG1-CY7111
CYPM1211-40LQXI	CY7112 EZ-PD™ PMG1-S2 prototyping kit	PMG1-CY7112
CYPM1311-48LQXI	CY7113 EZ-PD™ PMG1-S3 prototyping kit	PMG1-CY7113
CYPM1322-97BZXI	NA	CYPM1322-97BZXI

3 My first PMG1 MCU design using Eclipse IDE for ModusToolbox™ software

This section does the following:

- Demonstrate how to create a simple PMG1 MCU-based design, and build and program it on to the prototyping kit.
- Make it easy to learn PMG1 MCU design techniques and how to use the Eclipse IDE for ModusToolbox™ software.

3.1 Prerequisites

Before you get started, make sure that you have the appropriate kit for your PMG1 MCU product line and have installed the latest ModusToolbox™ software from **PMG1 MCU webpage**. You also need internet access to get the GitHub repositories during project creation.

See **Table 2** for the list of PMG1 MCU kits.

3.1.1 Hardware

The design is developed for CY7110 EZ-PD™ PMG1-S0 prototyping kit (PMG1-CY7110). However, you can build the projects for other prototyping kits. See the **Application development flow** section.

3.1.2 Software

- ModusToolbox™ software 2.3 or above

After installing the software, see the **ModusToolbox™ software user guide** to get an overview of the software.

3.2 Application development flow

These instructions are grouped into several sections. Each section is devoted to a phase of the application development workflow. The major sections are:

- **Part 1: Create a new application**
- **Part 2: View and modify the design**
- **Part 3: Write the firmware**
- **Part 4: Build the application**
- **Part 5: Program the device**
- **Part 6: Test your design**
- **Part 7: Debug the application**

The code described in the sections that follow has been tested on all the PMG1 MCU kits listed in **Table 2**.

3.3 About the design

This design uses the PMG1-S0 MCU to execute two tasks: UART communication and LED control. The project uses the UART to print a “Hello World” message to the serial port stream, and starts blinking the user LED on the kit.

3.4 Procedure

3.4.1 Part 1: Create a new application

This section takes you on a step-by-step guided tour of the new application creation process. It uses the "Hello World" code example and guides you through the design development stages, and programming.

If you are familiar with developing projects with ModusToolbox™ software, you can use the code example directly. It is a complete design, with all the firmware written for the supported kits. You can walk through the instructions in the code and observe how the steps are implemented.

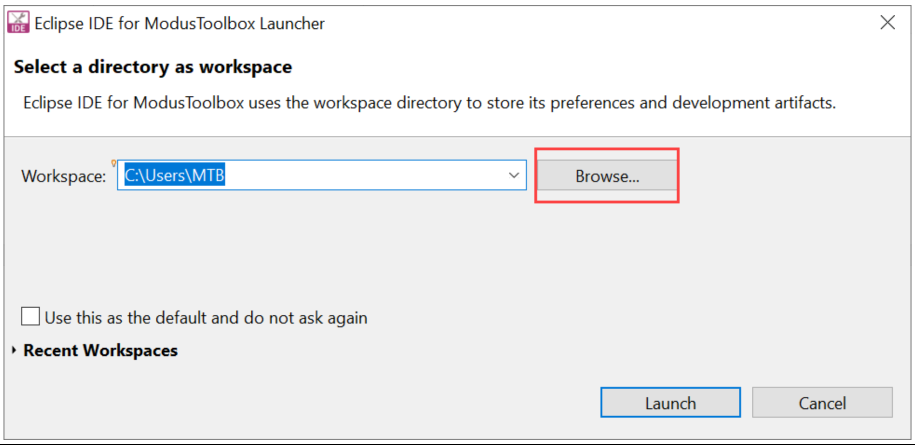
If you start from scratch and follow all the instructions in this section, you use the code example as a reference while following the instructions.

1. Launch Eclipse IDE for ModusToolbox™ software.

   Launch **Eclipse IDE for ModusToolbox™ software 2.3** to get started. Note that Eclipse IDE for ModusToolbox™ software needs access to the internet to successfully clone the template application onto your machine.

2. Select a new workspace.

   At launch, Eclipse IDE for ModusToolbox™ presents a dialog box to choose a directory for use as the workspace directory. The workspace directory is used to store workspace preferences and development artefacts. You can choose an existing empty directory by clicking the **Browse** button, as **Figure 8** shows. Alternatively, you can type in a directory name to be used as the workspace directory along with the complete path, and Eclipse IDE will create the directory for you.


Eclipse IDE for ModusToolbox Launcher

Select a directory as workspace

Eclipse IDE for ModusToolbox uses the workspace directory to store its preferences and development artifacts.

Workspace: C:\Users\MTB

Figure 8 Select/create a directory for the workspace

3. Create a new ModusToolbox™ application.
   a) Click New Application in the Start group of the Quick Panel.
   b) Alternatively, you can choose **File > New > ModusToolbox™ Application,** as **Figure 9** shows.

   The Eclipse IDE for ModusToolbox™ Application window appears.

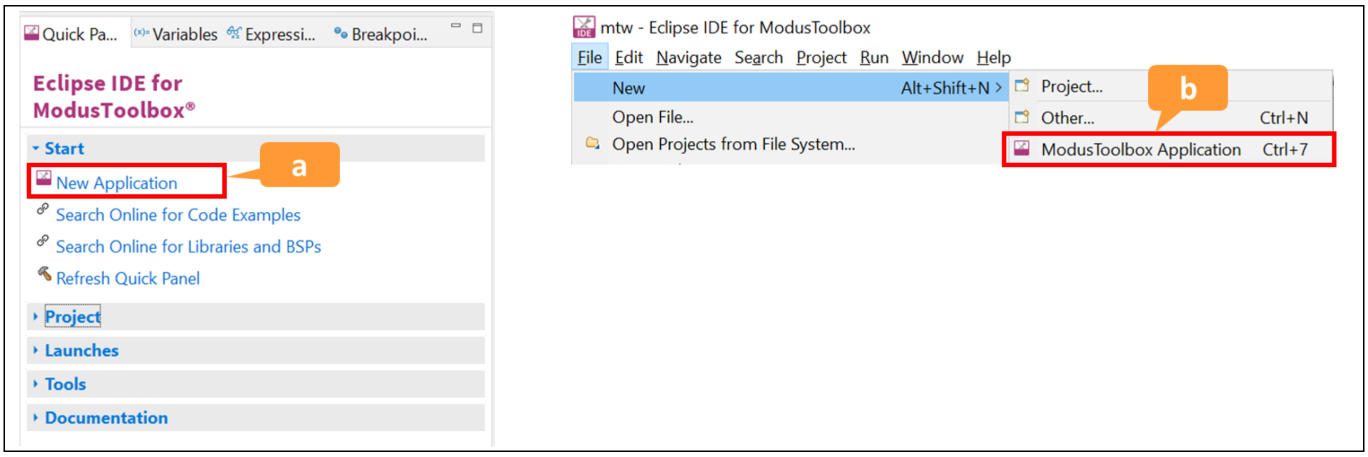
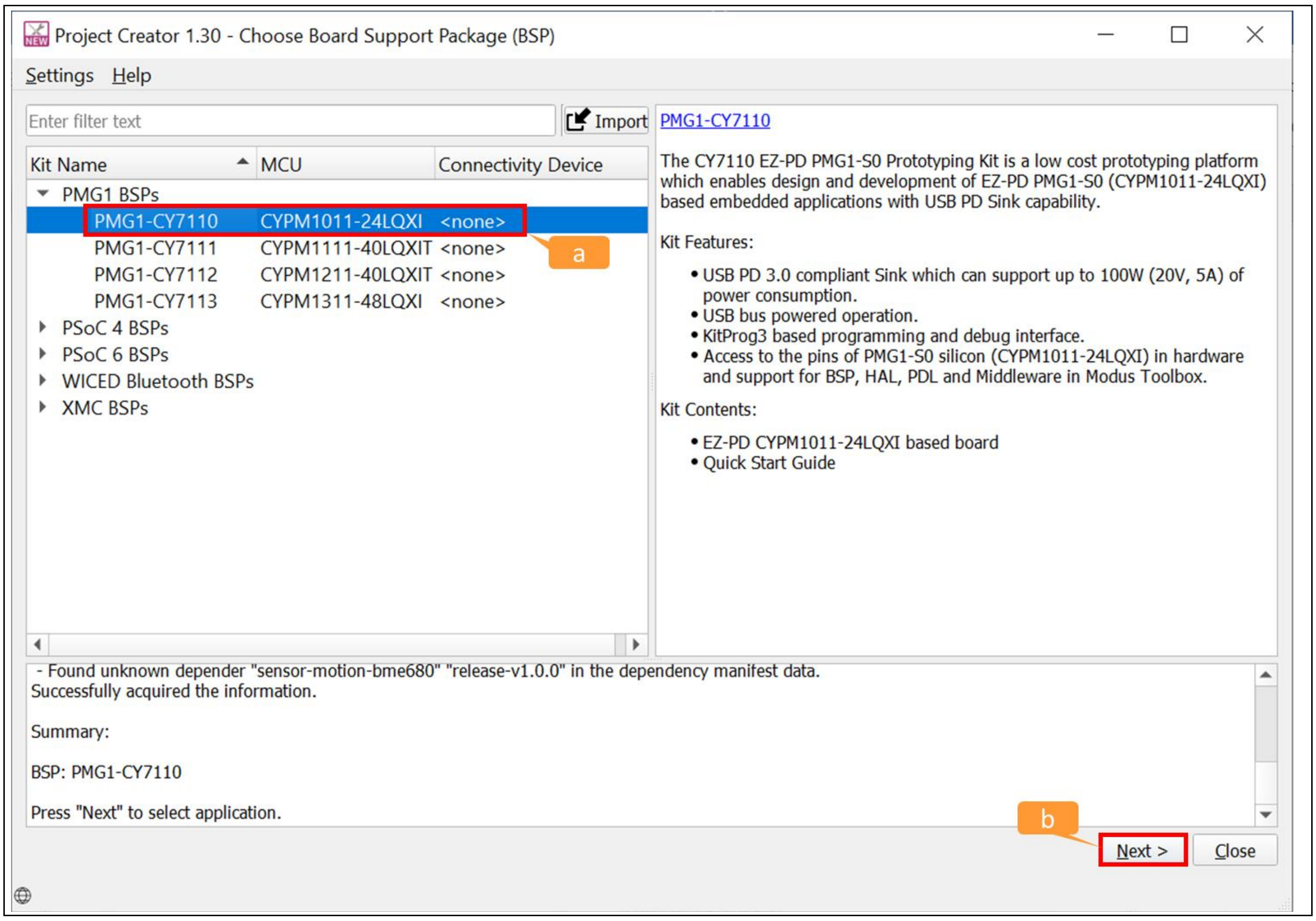

Quick Pa...
Variables
Expressi...
Breakpoi...
Eclipse IDE for ModusToolbox®
Start
New Application
a
Search Online for Code Examples
Search Online for Libraries and BSPs
Refresh Quick Panel
Project
Launches
Tools
Documentation
mtw - Eclipse IDE for ModusToolbox
File Edit Navigate Search Project Run Window Help
New Alt+Shift+N
Open File...
Open Projects from File System...
Project...
b
Other... Ctrl+N
ModusToolbox Application Ctrl+7

Figure 9 Create a new ModusToolbox™ software application

4. Select a target PMG1 MCU kit.
   a) In the **Choose Board Support Package (BSP)** dialog, choose the **Kit Name** that you have. As you are creating for PMG1-S0, choose PMG1-CY7110. See **Figure 10** for help with this step.
   b) Click **Next**.



Kit Name	MCU	Connectivity Device
PMG1 BSPs		
PMG1-CY7110	CYPM1011-24LQXI	<none>
PMG1-CY7111	CYPM1111-40LQXIT	<none>
PMG1-CY7112	CYPM1211-40LQXIT	<none>
PMG1-CY7113	CYPM1311-48LQXI	<none>
PSoC 4 BSPs		
PSoC 6 BSPs		
WICED Bluetooth BSPs		
XMC BSPs		

PMG1-CY7110

The CY7110 EZ-PD PMG1-S0 Prototyping Kit is a low cost prototyping platform which enables design and development of EZ-PD PMG1-S0 (CYPM1011-24LQXI) based embedded applications with USB PD Sink capability.

Kit Features:

- USB PD 3.0 compliant Sink which can support up to 100W (20V, 5A) of power consumption.
- USB bus powered operation.
- KitProg3 based programming and debug interface.
- Access to the pins of PMG1-S0 silicon (CYPM1011-24LQXI) in hardware and support for BSP, HAL, PDL and Middleware in Modus Toolbox.

Kit Contents:

- EZ-PD CYPM1011-24LQXI based board
- Quick Start Guide

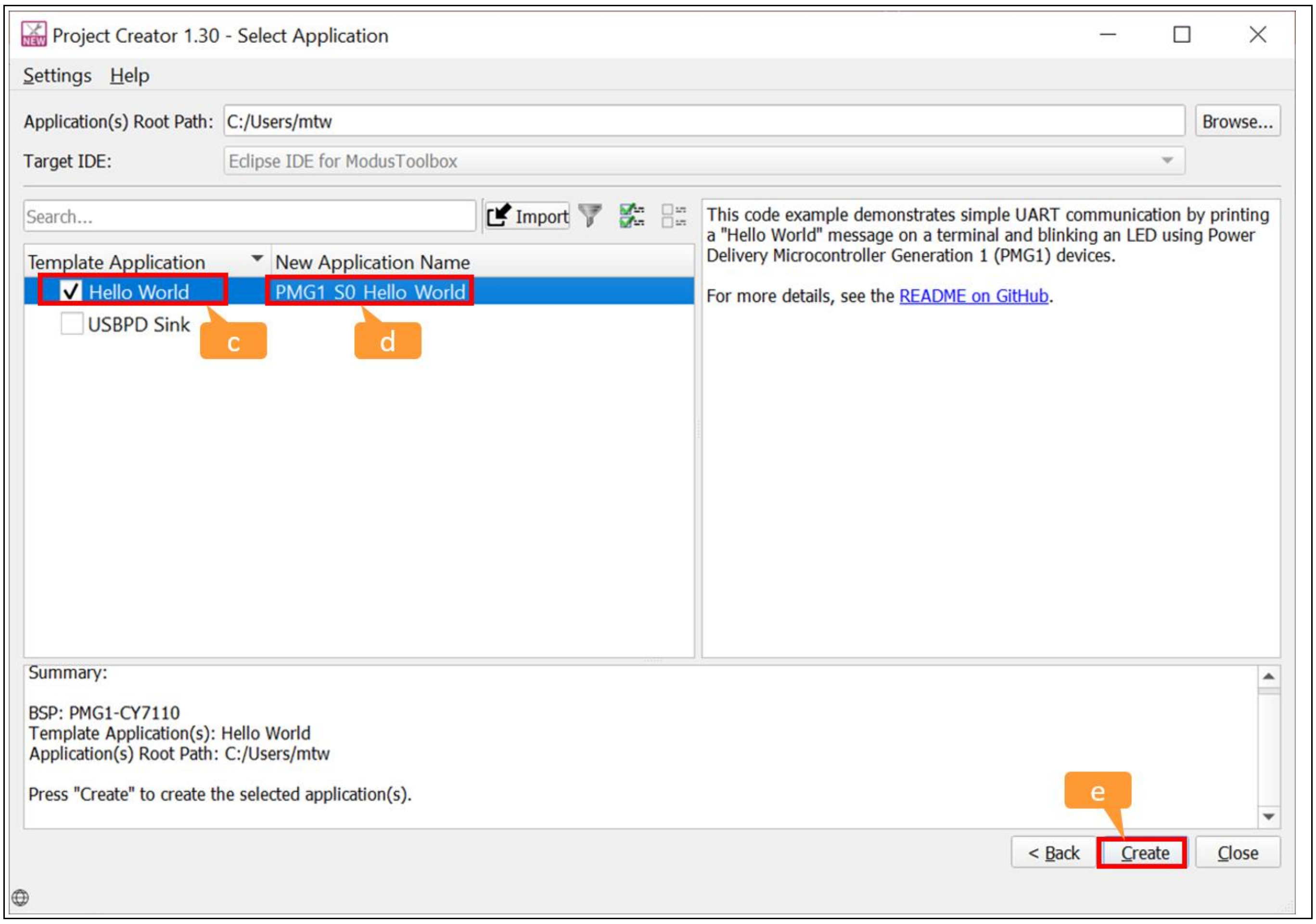
```
- Found unknown depender "sensor-motion-bme680" "release-v1.0.0" in the dependency manifest data.
Successfully acquired the information.

Summary:

BSP: PMG1-CY7110

Press "Next" to select application.
```

Figure 10 Choose the target hardware

   c) In the **Select Application** dialog, select **Hello World** template application, as **Figure 11** shows.

d) In the **Name** field, type in a name for the application, such as **PMG1_S0_Hello_World.** You can choose to leave the default name if you prefer.

e) Click **Create** to create the application and wait for the Project Creator to automatically close once the project is successfully created.



Application(s) Root Path: C:/Users/mtw

Target IDE: Eclipse IDE for ModusToolbox

Template Application	New Application Name
✓ Hello World	PMG1 S0 Hello World
USBPD Sink	

This code example demonstrates simple UART communication by printing a "Hello World" message on a terminal and blinking an LED using Power Delivery Microcontroller Generation 1 (PMG1) devices.

For more details, see the README on GitHub.

Summary:

BSP: PMG1-CY7110
Template Application(s): Hello World
Application(s) Root Path: C:/Users/mtw

Press "Create" to create the selected application(s).

Figure 11 **Choose the template 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 be opened 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 your Own BSP" section in the **ModusToolbox™ software user guide** (**Help** > **ModusToolbox™ General Documentation**).

3.4.2 Part 2: View and modify the design

Figure 12 shows the ModusToolbox™ software 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.

a) An application project contains a Makefile which is typically at the root folder. It has instructions on how to recreate the project. This file also contains the set of directives that the make tool uses to compile and link the application 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.

b) The *build* folder contains all the artefacts resulting from the make build of the project. The output files are organized by target BSPs.

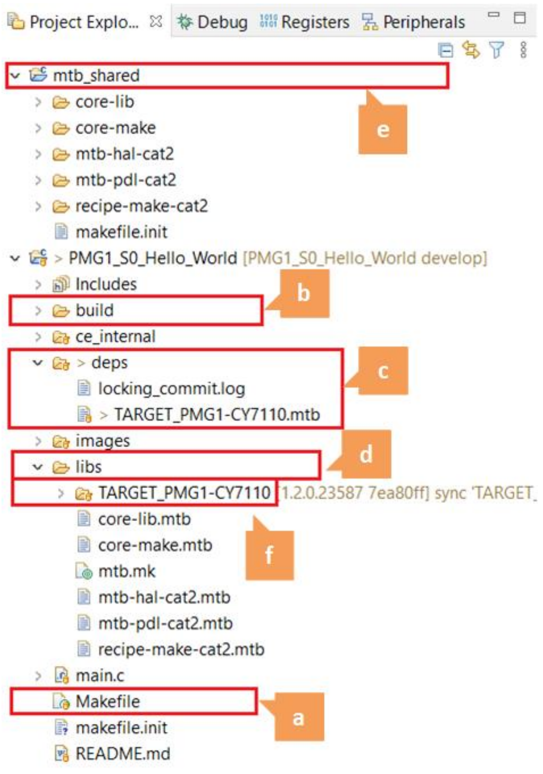


Figure 12 Project Explorer view

c) The *deps* folder contains *.mtb* files, which provide the location from which ModusToolbox™ software pulls the BSP/library that is directly referenced by the application. These files typically contain the GitHub location of the entire library. The *.mtb* files also contains a git Commit Hash or Tag that tells which version of the library is to be fetched and a path as to where the library should be stored.

d) The *libs* folder also contains *.mtb* files; these point to libraries that are included indirectly as a dependency of a BSP or another library. For each indirect dependency, the Library Manager places an *.mtb* file in this folder. These files have been populated based on the targets available in *deps* folder.

For example, using the BSP lib file *TARGET_PMG1-CY7110.mtb* populates the *libs* folder with the following *.mtb* files: **core-lib.mtb**, **core-make.mtb**, **mtb-hal-cat2.mtb**, **mtb-pdl-cat2.mtb**, **recipe-make-cat2.mtb**.

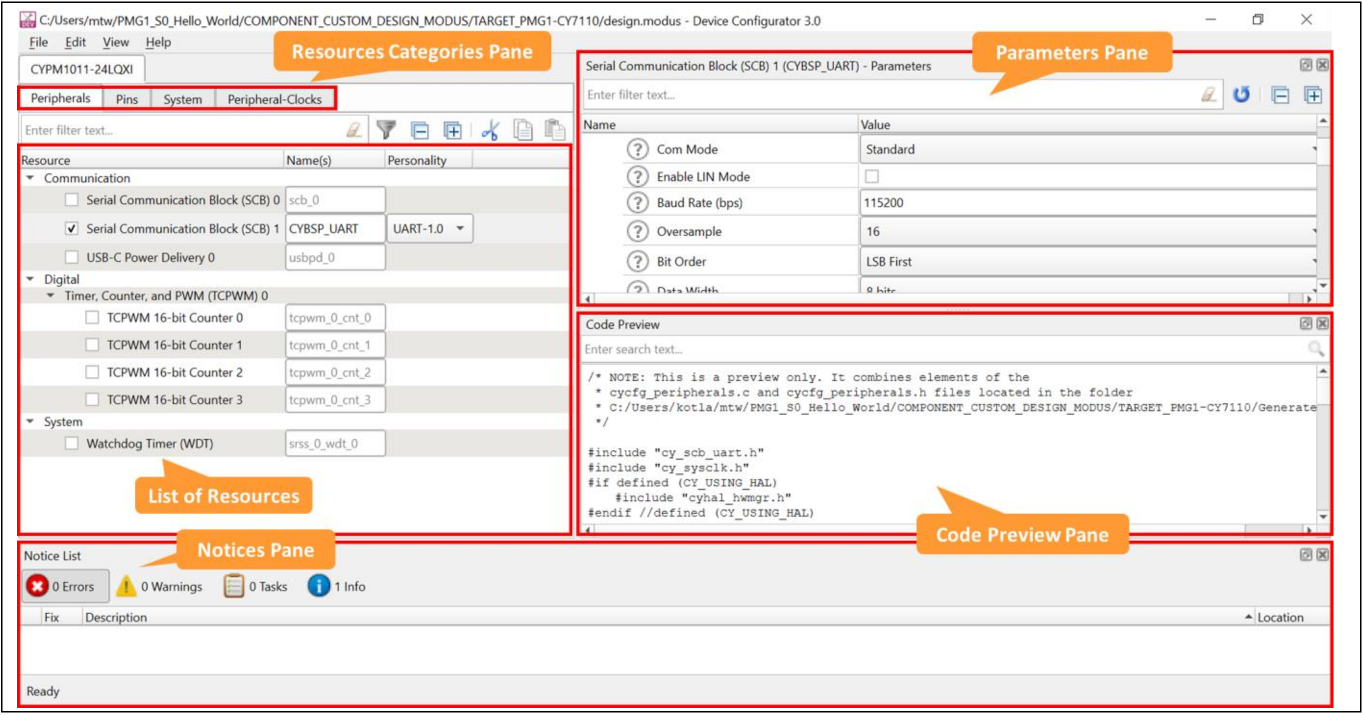
The libs folder contains *mtb.mk* file, which stores the relative paths of the all the libraries required by the application. The build system uses this file to find all the libraries required by the application.

e) 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.

f) The files provided by the BSP are listed under the *TARGET_x* folder. All the configuration files generated by the device and peripheral configurators are included in the *GeneratedSource* folder of the BSP and are prefixed with *cycfg_*. These files contain the design configuration as defined by the BSP. You can view and modify the design configuration by clicking the **Device Configurator** link in the Quick Panel. However, note that if you upgrade the BSP library to a newer version, the manual edits done to the *design.x* files are lost.

Of interest are the configuration files that are in the *COMPONENT_BSP_DESIGN_MODUS* folder. Click on the **Device Configurator** link in the Quick Panel. **Figure 13** shows the resulting window called the **Device Configurator** window. You can also double-click to open the other *design.x* files to open the respective configurators or click the corresponding links in the **Quick Panel**.



Resource	Name(s)	Personality
Communication		
☐ Serial Communication Block (SCB) 0	scb_0	
✓ Serial Communication Block (SCB) 1	CYBSP_UART	UART-1.0
☐ USB-C Power Delivery 0	usbpd_0	
Digital		
Timer, Counter, and PWM (TCPWM) 0		
☐ TCPWM 16-bit Counter 0	tcpwm_0_cnt_0	
☐ TCPWM 16-bit Counter 1	tcpwm_0_cnt_1	
☐ TCPWM 16-bit Counter 2	tcpwm_0_cnt_2	
☐ TCPWM 16-bit Counter 3	tcpwm_0_cnt_3	
System		
☐ Watchdog Timer (WDT)	srss_0_wdt_0	

Serial Communication Block (SCB) 1 (CYBSP_UART) - Parameters

Name	Value
Com Mode	Standard
Enable LIN Mode	☐
Baud Rate (bps)	115200
Oversample	16
Bit Order	LSB First
Data Width	8 bits

Code Preview

```
/* NOTE: This is a preview only. It combines elements of the
 * cycfg_peripherals.c and cycfg_peripherals.h files located in the folder
 * C:/Users/kotla/mtw/PMG1_S0_Hello_World/COMPONENT_CUSTOM_DESIGN_MODUS/TARGET_PMG1-CY7110/Generate
 */

#include "cy_scb_uart.h"
#include "cy_sysclk.h"
#if defined (CY_USING_HAL)
    #include "cyhal_hwmgr.h"
#endif //defined (CY_USING_HAL)
```

Notice List

0 Errors 0 Warnings 0 Tasks 1 Info

Fix	Description	Location

Ready

Figure 13 design.modus overview

The **Device Configurator** window provides a **Resources Categories** pane. Here, can you can choose between different resources available in the device such as peripherals, pins, and clocks from the **List of Resources**.

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 **Name** is your 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)

The **Parameters** pane is where you 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. Any errors, warnings, and information messages arising out of the configuration are displayed in the **Notices** pane.

At this point in the development process, you are ready to add any required middleware to the design. **Figure 28** shows an example of adding a middleware. For the "Hello World" example project, no additional middleware needs to be added.

The configuration of the debug UART peripheral, pins, and system clocks can be done directly in the code using the function APIs provided by BSP. See **Part 3: Write the firmware**.

3.4.3 Part 3: Write the firmware

At this point in the development process, you have created an application, with the assistance of a template application. In this section, you write the firmware that implements the design functionality.

If you are working from scratch, you can copy the source code provided in this section to the *main.c* of the application project. If you are using the PMG1 MCU 'Hello World' code example, all the required files are already in the application.

3.4.3.1 Firmware flow

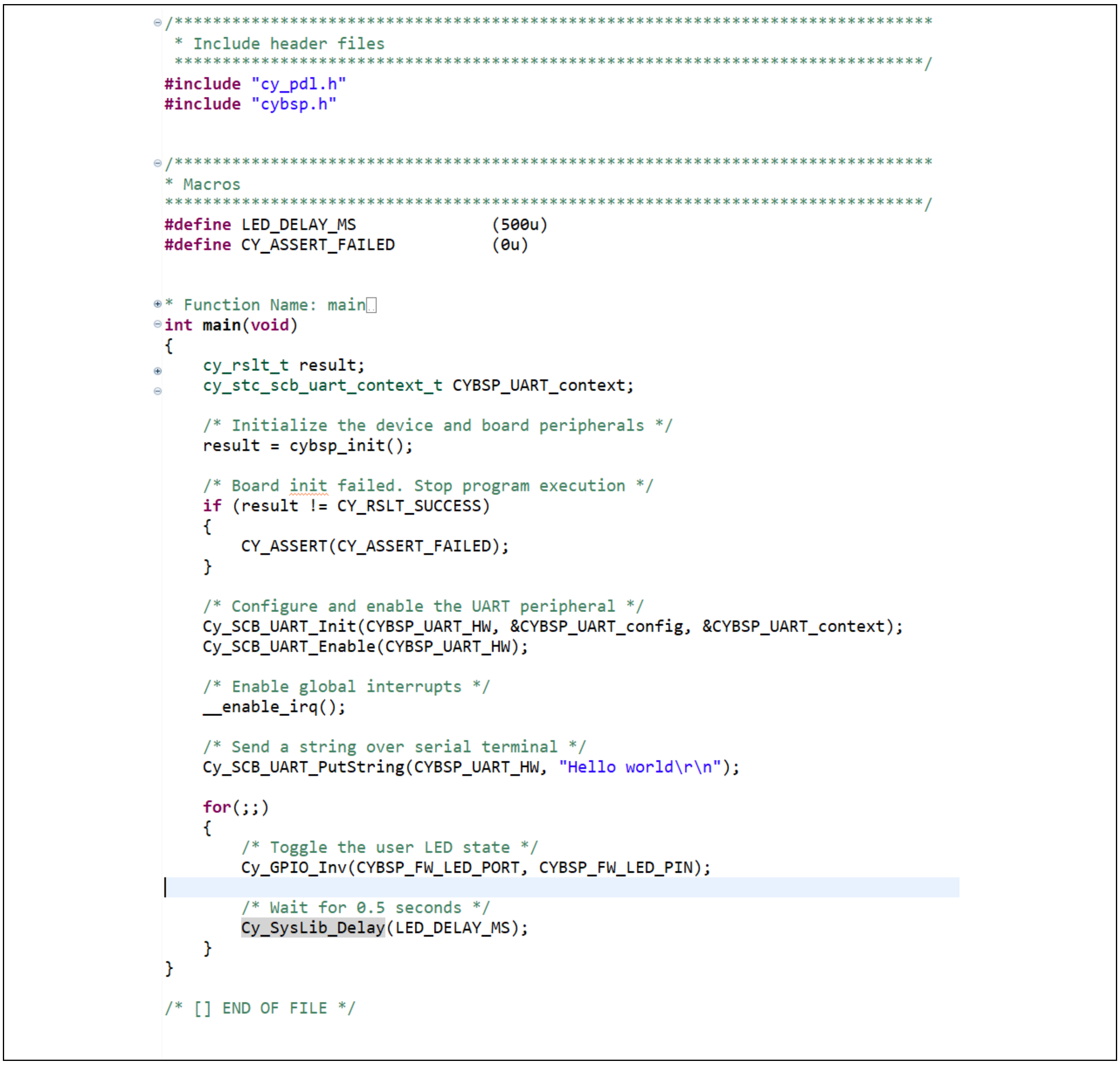
You now examine the code in the *main.c* file of the application. **Figure 14** shows the firmware flowchart.

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

Copy the following code snippet to *main.c* of your application project.



```
/*******************************************************************************
 * 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 */
```

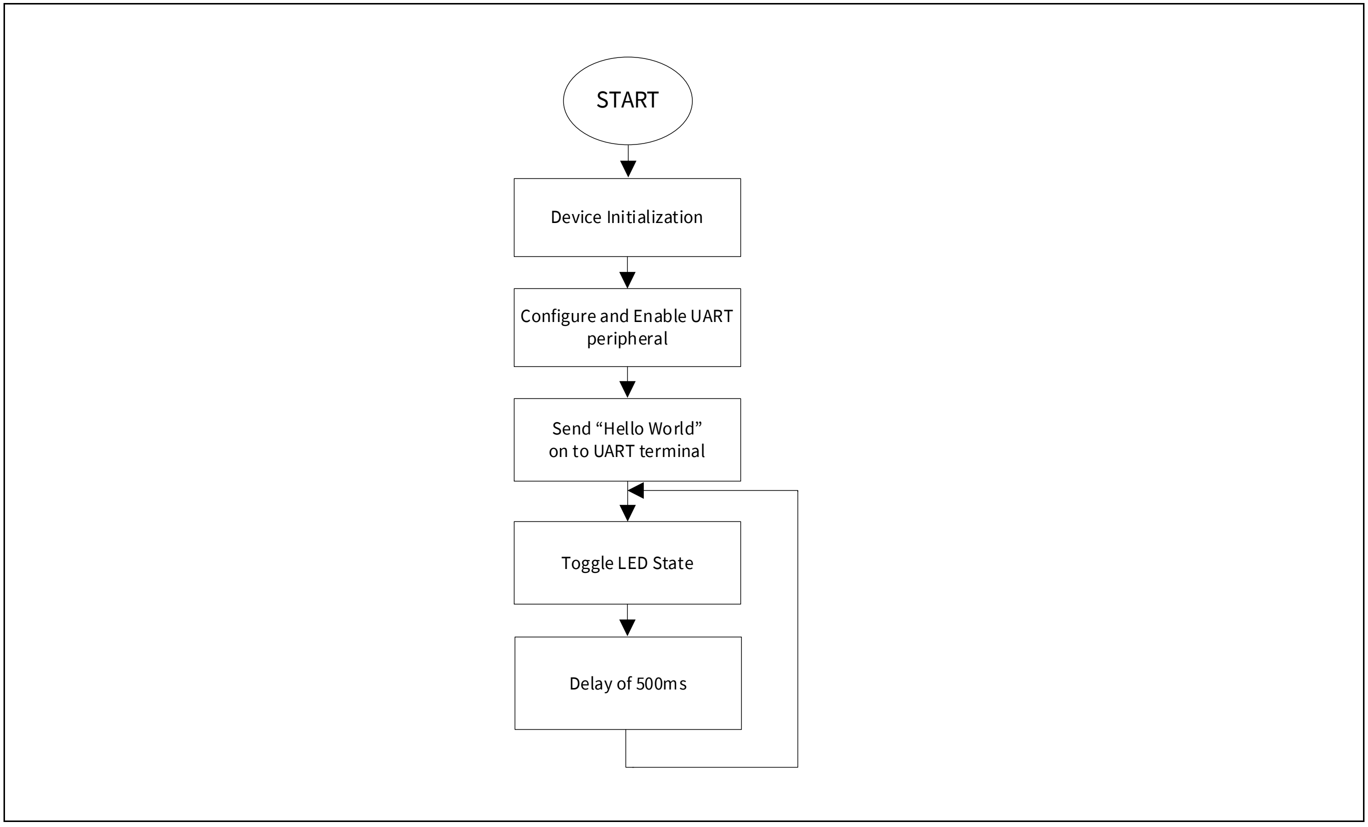

START
Device Initialization
Configure and Enable UART peripheral
Send “Hello World” on to UART terminal
Toggle LED State
Delay of 500ms

Figure 14 Firmware flowchart

This completes the summary of how the firmware works in the code example. Feel free to explore the source files for a deeper understanding.

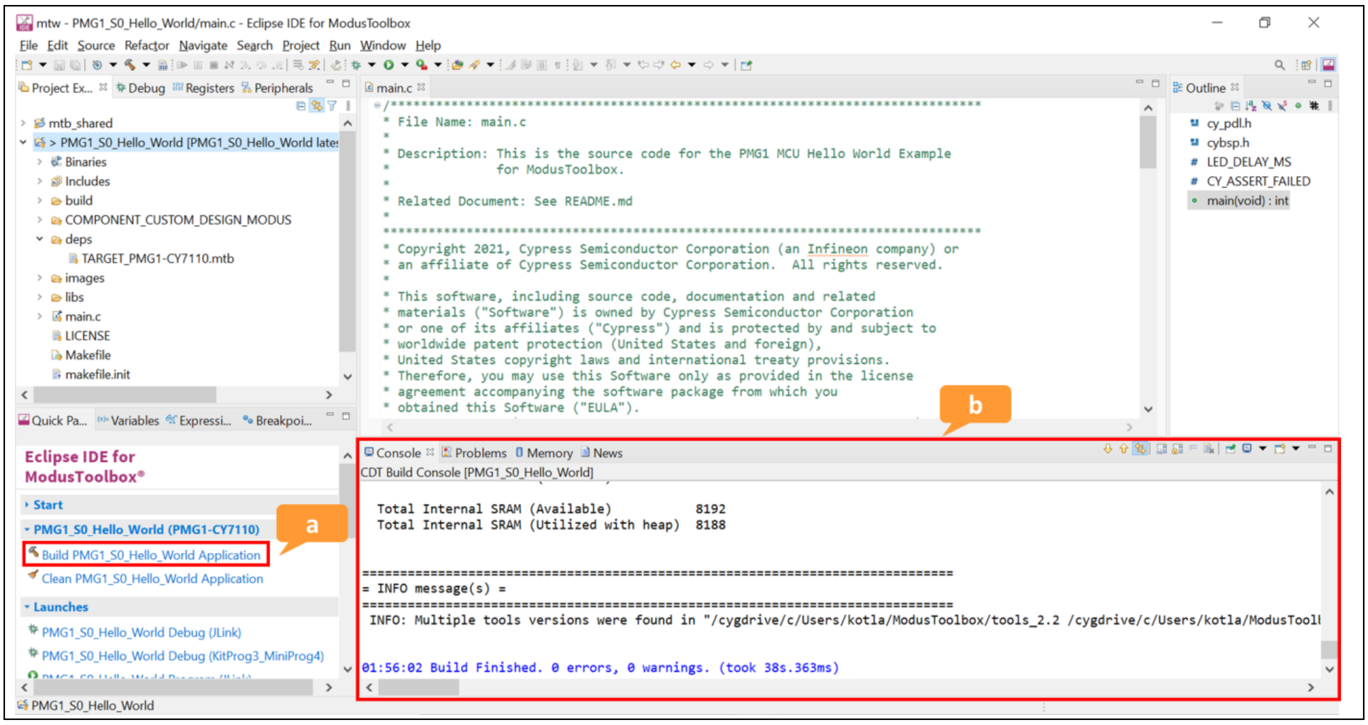
3.4.4 Part 4: Build the application

This section shows how to build the application.

1. Build the application.

   a) Select the application project in the Project Explorer window and click on the **Build <name> Application** shortcut under the **<name>** group in the Quick Panel. It selects the **Debug** build configuration and compiles/links all projects that constitute the application.

   b) The **Console** view lists the results of the build operation, as **Figure 15** shows.


mtw - PMG1_S0_Hello_World/main.c - Eclipse IDE for ModusToolbox
File Edit Source Refactor Navigate Search Project Run Window Help
Project Ex... Debug Registers Peripherals
mtb_shared
> PMG1_S0_Hello_World [PMG1_S0_Hello_World late
Binaries
Includes
build
COMPONENT_CUSTOM_DESIGN_MODUS
deps
TARGET_PMG1-CY7110.mtb
images
libs
main.c
LICENSE
Makefile
makefile.init
main.c
Outline
cy_pdl.h
cybsp.h
LED_DELAY_MS
CY_ASSERT_FAILED
main(void) : int
Quick Pa... Variables Expressi... Breakpoi...
Eclipse IDE for ModusToolbox®
Start
PMG1_S0_Hello_World (PMG1-CY7110)
Build PMG1_S0_Hello_World Application
Clean PMG1_S0_Hello_World Application
Launches
PMG1_S0_Hello_World Debug (JLink)
PMG1_S0_Hello_World Debug (KitProg3_MiniProg4)
PMG1_S0_Hello_World
a
b
Console Problems Memory News
CDT Build Console [PMG1_S0_Hello_World]

```
/*****************************************************************************
* File Name: main.c
*
* Description: This is the source code for the PMG1 MCU Hello World Example
*              for ModusToolbox.
*
* Related Document: See README.md
*
*****************************************************************************
* Copyright 2021, Cypress Semiconductor Corporation (an Infineon company) or
* an affiliate of Cypress Semiconductor Corporation.  All rights reserved.
*
* This software, including source code, documentation and related
* materials ("Software") is owned by Cypress Semiconductor Corporation
* or one of its affiliates ("Cypress") and is protected by and subject to
* worldwide patent protection (United States and foreign),
* United States copyright laws and international treaty provisions.
* Therefore, you may use this Software only as provided in the license
* agreement accompanying the software package from which you
* obtained this Software ("EULA").
```

```
 Total Internal SRAM (Available)          8192
 Total Internal SRAM (Utilized with heap)  8188


==============================================================================
= INFO message(s) =
==============================================================================
 INFO: Multiple tools versions were found in "/cygdrive/c/Users/kotla/ModusToolbox/tools_2.2 /cygdrive/c/Users/kotla/ModusTool

01:56:02 Build Finished. 0 errors, 0 warnings. (took 38s.363ms)
```

Figure 15 Build the application

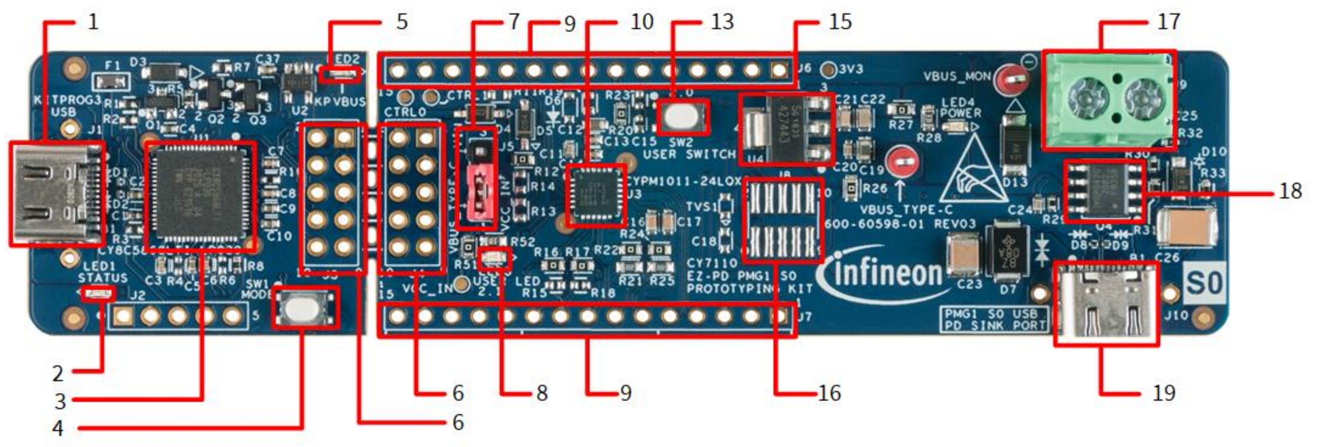
If you encounter errors, try the following options:

- 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 <application name>' after which try to build the project again.

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

3.4.5 Part 5: Program the device

This section explains how to program the device. Refer **Figure 16** for CY7110 EZ-PD™ PMG1-S0 prototyping kit. See the 'Kits' 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 16 Choose CY7110 EZ-PD™ PMG1-S0 prototyping kit

ModusToolbox™ software uses the OpenOCD protocol to program and debug applications on PMG1 MCU devices. For ModusToolbox™ software to identify the device on the kit, the kit must be running KitProg3.

If you are using a PMG1 MCU prototyping kit with a built-in programmer, follow steps 1 and 2.

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

1. Make the following hardware connections on the prototyping kit:
   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.*

2. Program the board:
   a) In ModusToolbox™ software, select the application project in the Project Explorer.
   b) In the Quick Panel, scroll down, and click **\<Application Name> Program (KitProg3_MiniProg4)**.

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.

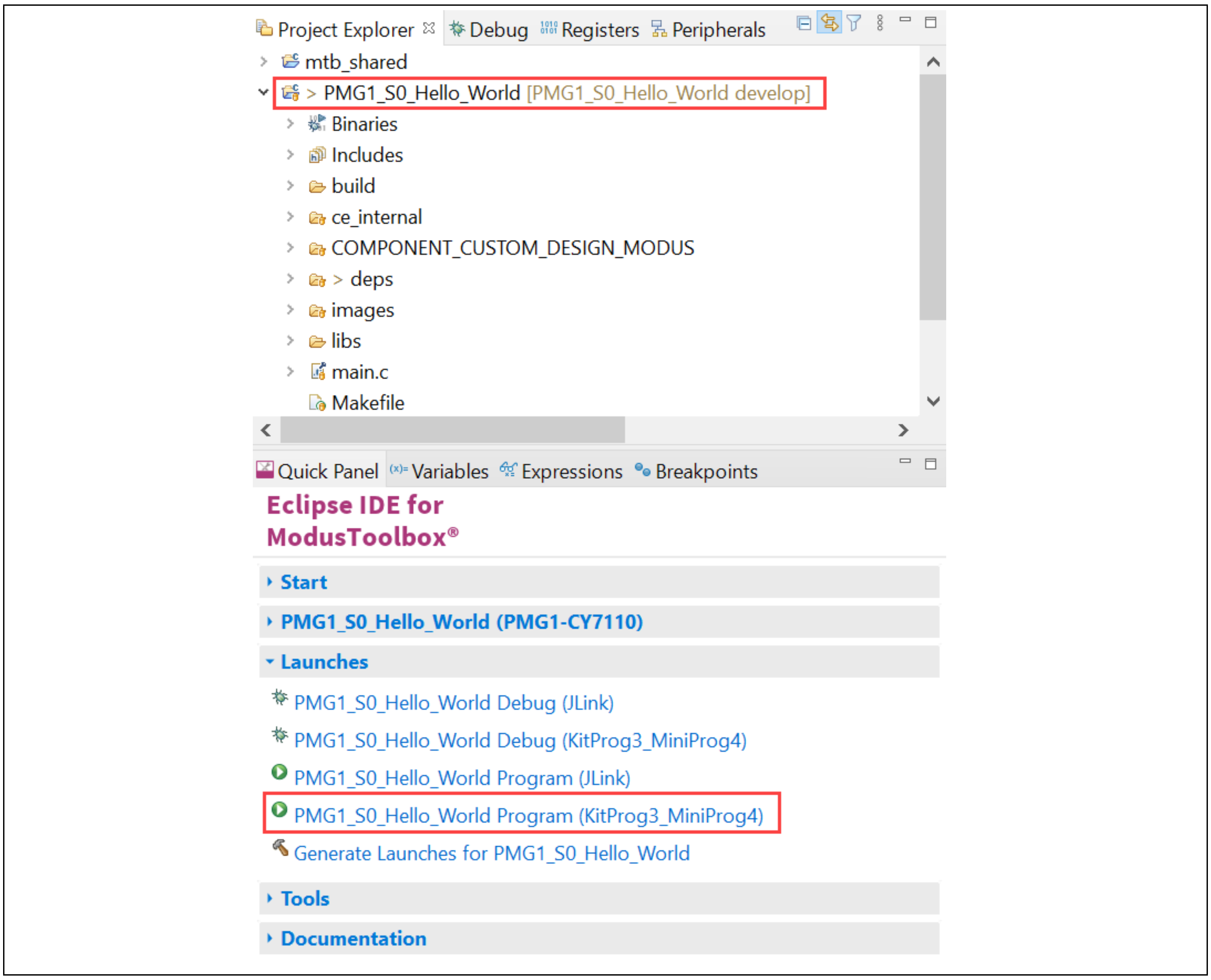
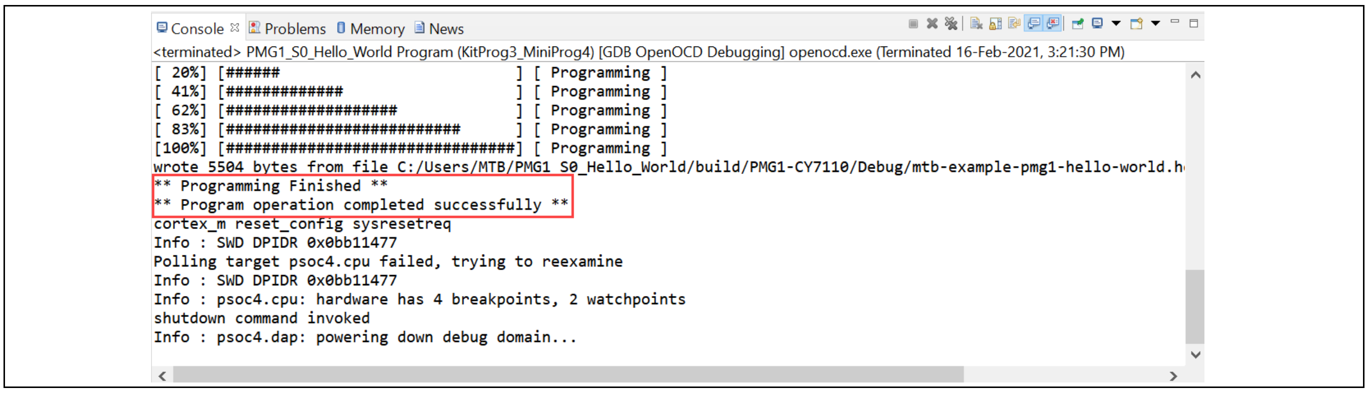

Project Explorer
Debug
Registers
Peripherals
mtb_shared
> PMG1_S0_Hello_World [PMG1_S0_Hello_World develop]
Binaries
Includes
build
ce_internal
COMPONENT_CUSTOM_DESIGN_MODUS
> deps
images
libs
main.c
Makefile
Quick Panel
Variables
Expressions
Breakpoints
Eclipse IDE for ModusToolbox®
Start
PMG1_S0_Hello_World (PMG1-CY7110)
Launches
PMG1_S0_Hello_World Debug (JLink)
PMG1_S0_Hello_World Debug (KitProg3_MiniProg4)
PMG1_S0_Hello_World Program (JLink)
PMG1_S0_Hello_World Program (KitProg3_MiniProg4)
Generate Launches for PMG1_S0_Hello_World
Tools
Documentation

Figure 17 Programming an application to a device

3. The **Console** view in the ModusToolbox™ software lists the results of the programming operation, as **Figure 18** shows.



```
<terminated> PMG1_S0_Hello_World Program (KitProg3_MiniProg4) [GDB OpenOCD Debugging] openocd.exe (Terminated 16-Feb-2021, 3:21:30 PM)
[ 20%] [######                          ] [ Programming ]
[ 41%] [#############                   ] [ Programming ]
[ 62%] [###################             ] [ Programming ]
[ 83%] [##########################      ] [ Programming ]
[100%] [################################] [ Programming ]
wrote 5504 bytes from file C:/Users/MTB/PMG1_S0_Hello_World/build/PMG1-CY7110/Debug/mtb-example-pmg1-hello-world.h
** Programming Finished **
** Program operation completed successfully **
cortex_m reset_config sysresetreq
Info : SWD DPIDR 0x0bb11477
Polling target psoc4.cpu failed, trying to reexamine
Info : SWD DPIDR 0x0bb11477
Info : psoc4.cpu: hardware has 4 breakpoints, 2 watchpoints
shutdown command invoked
Info : psoc4.dap: powering down debug domain...
```

Figure 18 Console – programming results

3.4.6 Part 6: Test your design

This section explains how to 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.

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:

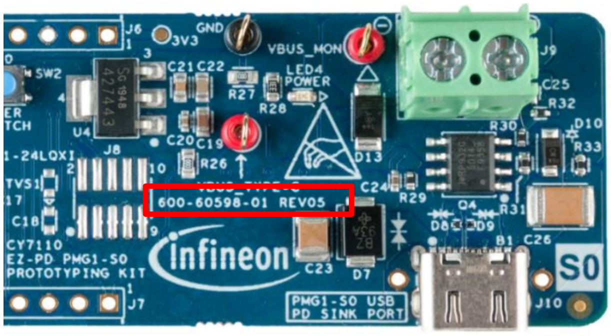
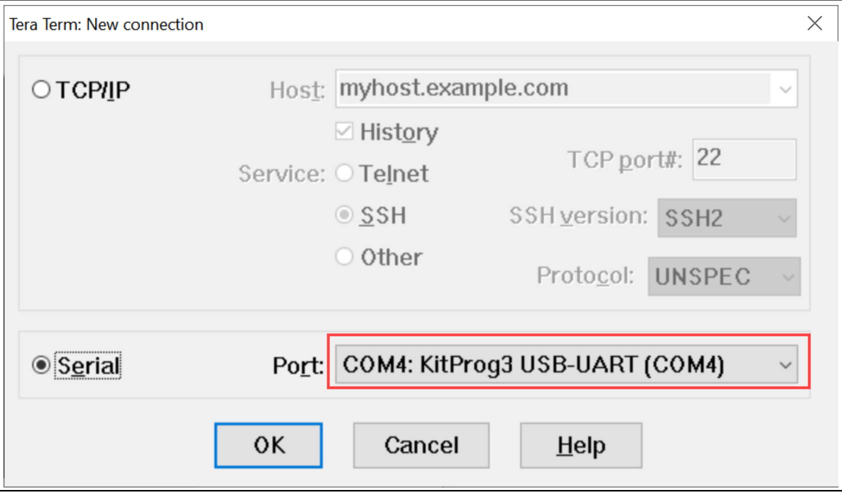


Figure 19 Identifying PMG1 MCU kit board revision number

The UART connection between KitProg3 and PMG1-S0 is pre-established by default on the newer revisions of the kit boards.

   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 amber.

2. Open a terminal program and select the KitProg3 COM port. 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.

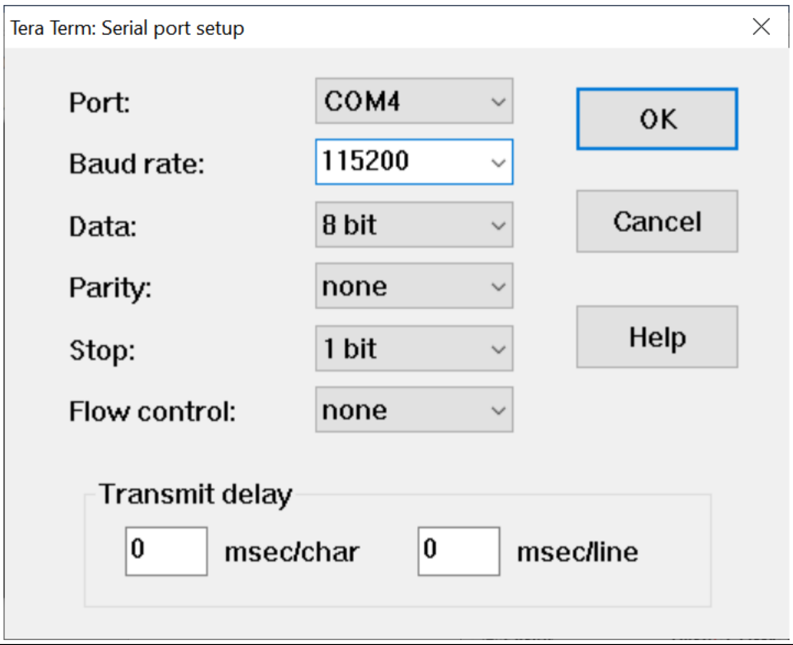

Tera Term: New connection

TCP/IP Host: myhost.example.com
History
Service: Telnet
SSH
Other
TCP port#: 22
SSH version: SSH2
Protocol: UNSPEC

Serial Port: COM4: KitProg3 USB-UART (COM4)

Figure 20 Selecting the KitProg3 COM port in Tera Term

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

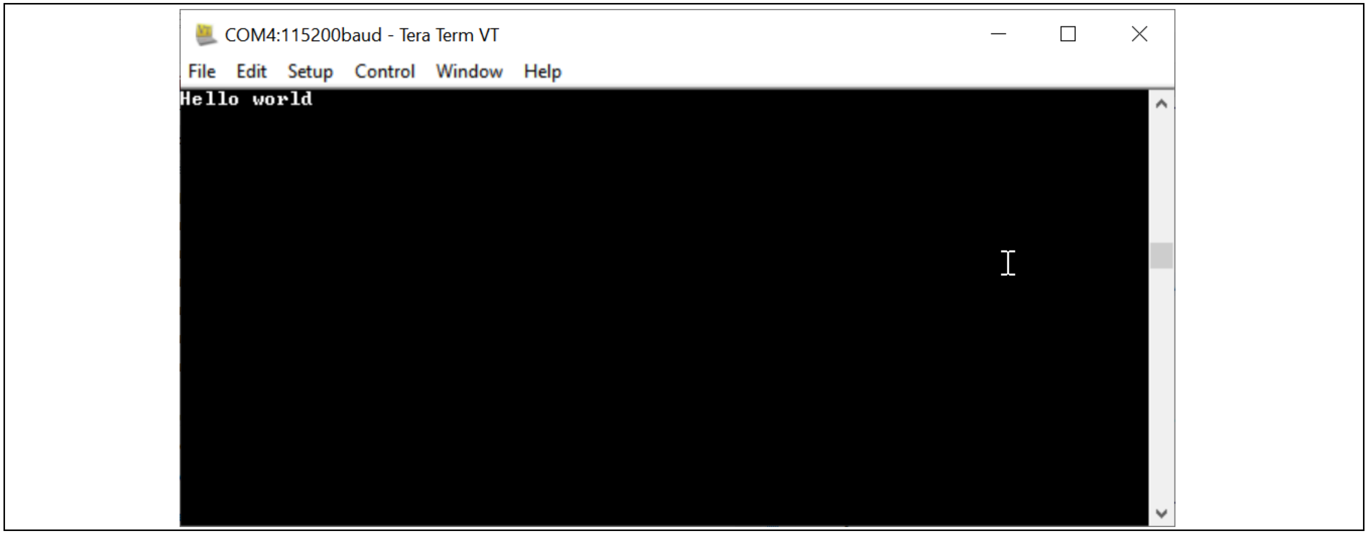

Tera Term: Serial port setup

Port: COM4
Baud rate: 115200
Data: 8 bit
Parity: none
Stop: 1 bit
Flow control: none

Transmit delay
0 msec/char 0 msec/line

Figure 21 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.


COM4:115200baud - Tera Term VT
File Edit Setup Control Window Help

```
Hello world
```

Figure 22 “Hello World” in terminal output

6. Confirm that the Power LED (LED4) glows green and User LED (LED3) blinks green.

3.4.7 Part 7: 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 similar to programming mode:

- 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.

3.4.7.1 Debugging through the KitProg3 interface

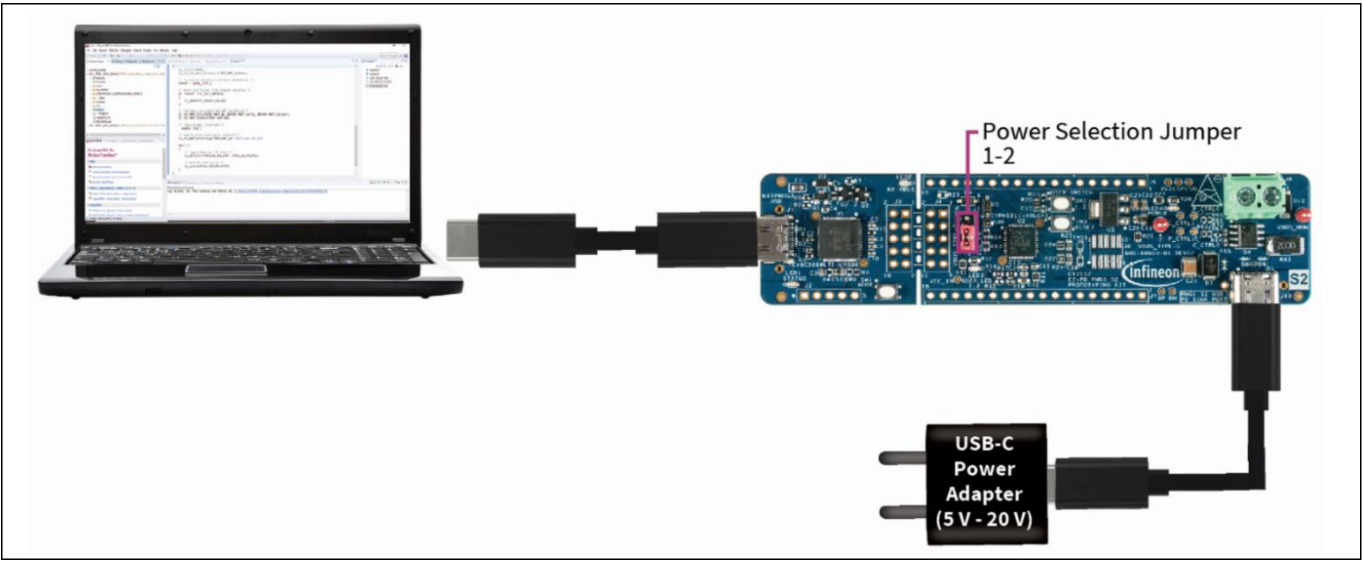


Figure 23 Debugging through KitProg3

Do the following to debug an application on the CY7111 and CY7112 prototyping kits over KitProg3 interface from Eclipse IDE for ModusToolbox™ software.

1. Open the Eclipse IDE for ModusToolbox™ software and select the PMG1 MCU project on the host PC. For details, see the **Eclipse IDE for ModusToolbox™ software user guide**.
2. Place the jumper shunt on pins 1–2 of the power section jumper (J5) to configure the kit in operational mode.
3. 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.
4. Connect the kit to the host PC through programming KitProg3 USB Type-C Port. Ensure that LED1 and LED2 glow amber. LED2, which is KitProg3 Power LED, indicates that the KitProg3 module is powered. LED1, which is the status LED, indicates the programming/debug mode and 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.
6. 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.

Do the following to debug an application on the CY7110 prototyping kits over KitProg3 interface from Eclipse IDE for ModusToolbox™ software.

1. Open the Eclipse IDE for ModusToolbox™ software and select the PMG1 MCU project on the host PC or laptop. For details, see the **Eclipse IDE for ModusToolbox™ software user guide**.
2. Power selection jumper (J5) shunt needs to be placed on pins 2–3 of the power selection jumper (J5) to enable ModusToolbox™ software to acquire the target (PMG1-S0) in power cycle mode.
3. 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.
4. Connect the kit to the host PC through programming KitProg3 USB Type-C Port. Ensure that LED1 and LED2 glow amber. LED2, which is KitProg3 Power LED, indicates that the KitProg3 module is powered. LED1, which is the status LED, indicates the programming/debug mode and status, and is ON when KitProg3 is powered.
5. Go to the **Quick Panel** tab. Click **<Application name> Program (KitProg3_MiniProg4)** from the **Launches** section to download the firmware into the kit as shown in **Figure 24**.
6. After program download completed, change the jumper to short 1–2 of the power section jumper (J5) to configure the kit in operational mode for debugging.
7. Go to **Run** menu, click on **Debug Configurations**, as shown in in **Figure 25**.
8. Under GDB OpenOCD Debugging, select **<Application name> Attach (KitProg3_MiniProg4)** and click on **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™ software user guide**.

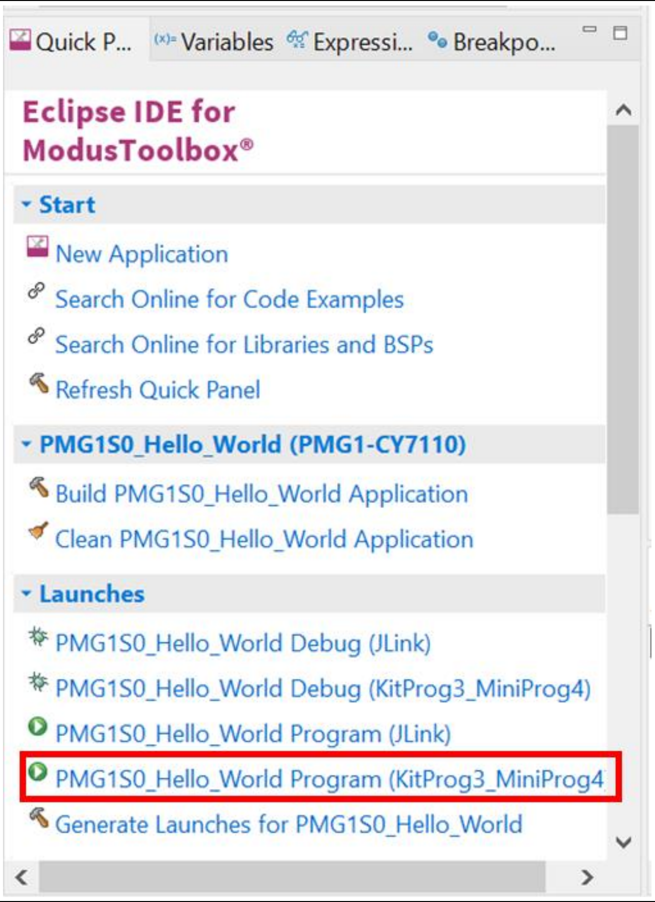
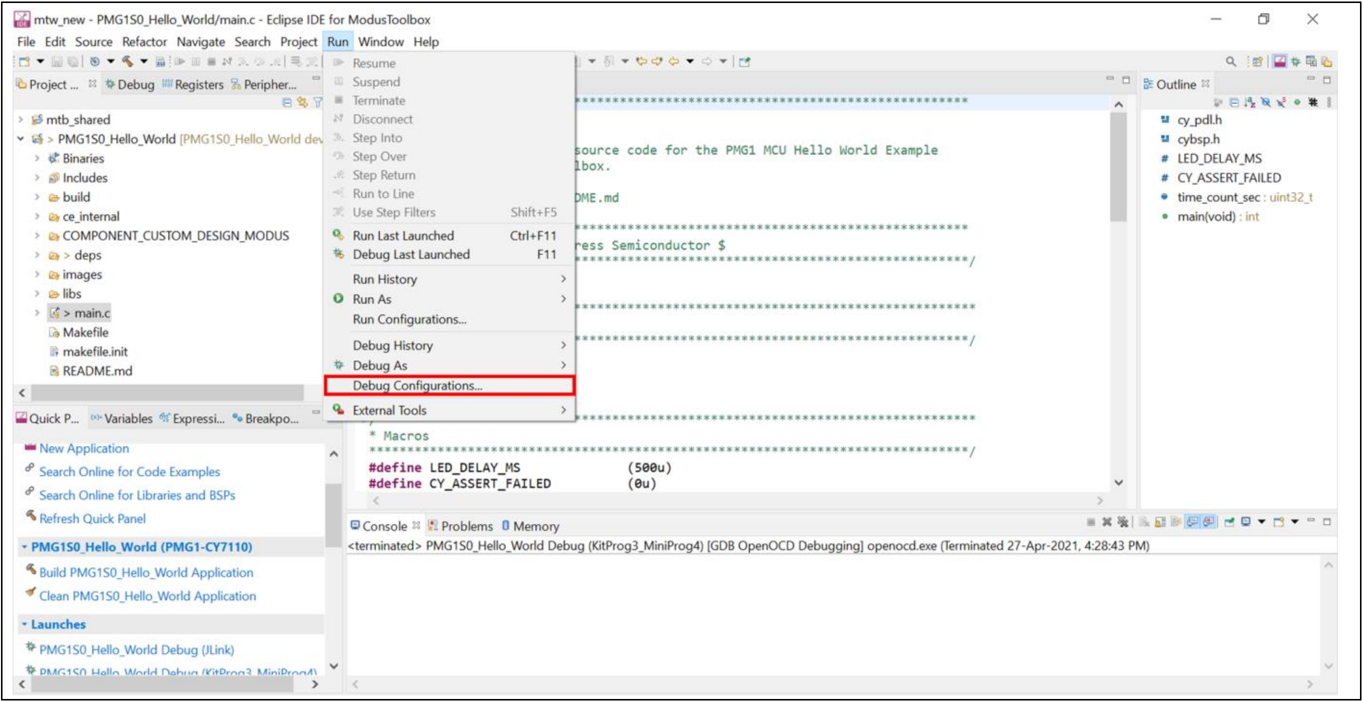

Quick P...
Variables
Expressi...
Breakpo...
Eclipse IDE for ModusToolbox®
Start
New Application
Search Online for Code Examples
Search Online for Libraries and BSPs
Refresh Quick Panel
PMG1S0_Hello_World (PMG1-CY7110)
Build PMG1S0_Hello_World Application
Clean PMG1S0_Hello_World Application
Launches
PMG1S0_Hello_World Debug (JLink)
PMG1S0_Hello_World Debug (KitProg3_MiniProg4)
PMG1S0_Hello_World Program (JLink)
PMG1S0_Hello_World Program (KitProg3_MiniProg4
Generate Launches for PMG1S0_Hello_World

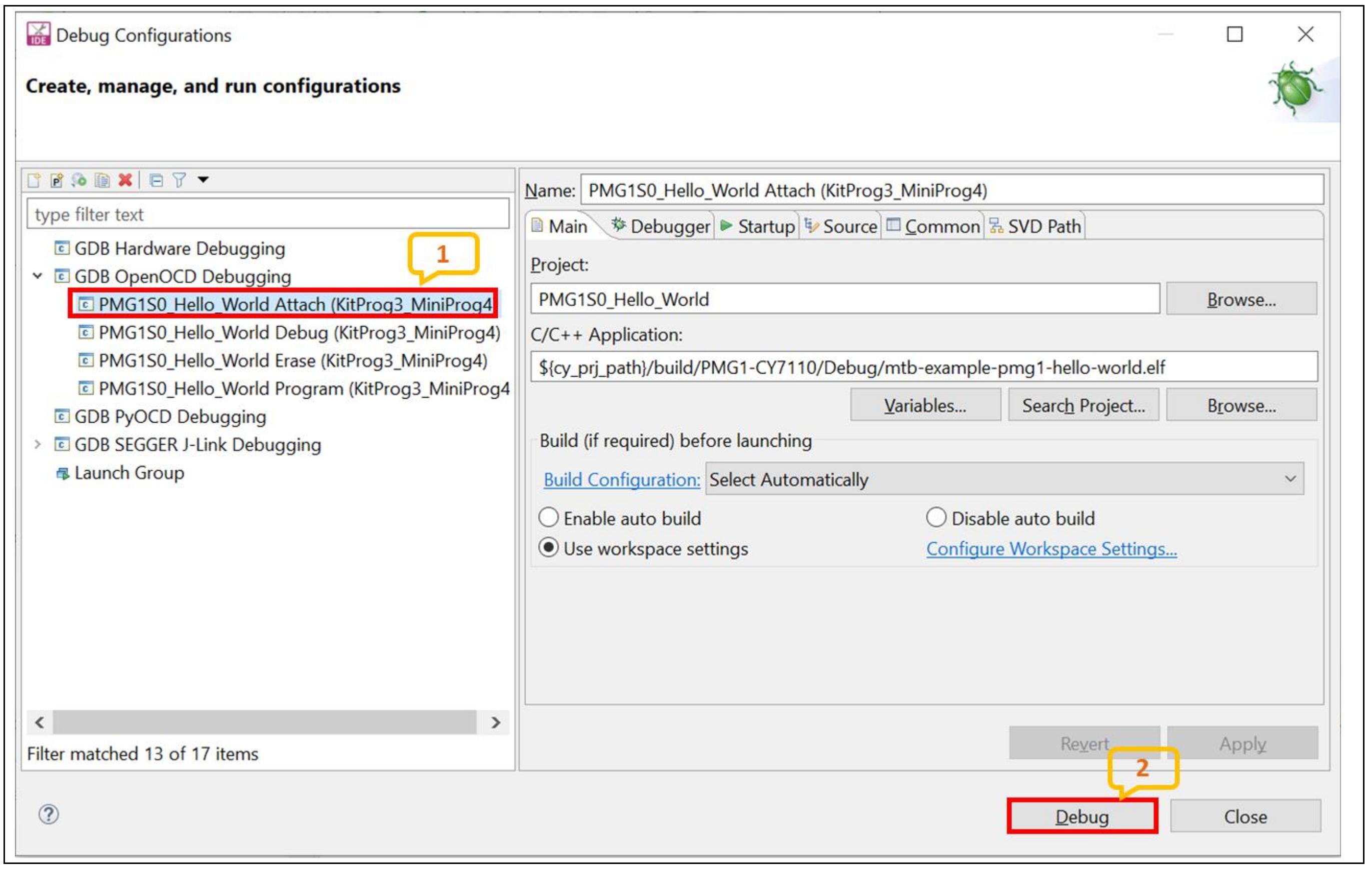
Figure 24 Quick panel and debug options


mtw_new - PMG1S0_Hello_World/main.c - Eclipse IDE for ModusToolbox
File Edit Source Refactor Navigate Search Project Run Window Help
Resume
Suspend
Terminate
Disconnect
Step Into
Step Over
Step Return
Run to Line
Use Step Filters Shift+F5
Run Last Launched Ctrl+F11
Debug Last Launched F11
Run History
Run As
Run Configurations...
Debug History
Debug As
Debug Configurations...
External Tools
Project ... Debug Registers Peripher...
mtb_shared
PMG1S0_Hello_World [PMG1S0_Hello_World dev
Binaries
Includes
build
ce_internal
COMPONENT_CUSTOM_DESIGN_MODUS
deps
images
libs
main.c
Makefile
makefile.init
README.md
Outline
cy_pdl.h
cybsp.h
LED_DELAY_MS
CY_ASSERT_FAILED
time_count_sec : uint32_t
main(void) : int
Quick P... Variables Expressi... Breakpo...
New Application
Search Online for Code Examples
Search Online for Libraries and BSPs
Refresh Quick Panel
PMG1S0_Hello_World (PMG1-CY7110)
Build PMG1S0_Hello_World Application
Clean PMG1S0_Hello_World Application
Launches
PMG1S0_Hello_World Debug (JLink)
PMG1S0_Hello_World Debug (KitProg3_MiniProg4)
Console Problems Memory

```
source code for the PMG1 MCU Hello World Example
lbox.

DME.md

ress Semiconductor $

* Macros

#define LED_DELAY_MS            (500u)
#define CY_ASSERT_FAILED        (0u)
```

```
<terminated> PMG1S0_Hello_World Debug (KitProg3_MiniProg4) [GDB OpenOCD Debugging] openocd.exe (Terminated 27-Apr-2021, 4:28:43 PM)
```

Figure 25 Selecting Debug configurations from the Run menu



Debug Configurations

Create, manage, and run configurations

type filter text

- GDB Hardware Debugging
- GDB OpenOCD Debugging
  - PMG1S0_Hello_World Attach (KitProg3_MiniProg4)
  - PMG1S0_Hello_World Debug (KitProg3_MiniProg4)
  - PMG1S0_Hello_World Erase (KitProg3_MiniProg4)
  - PMG1S0_Hello_World Program (KitProg3_MiniProg4
- GDB PyOCD Debugging
- GDB SEGGER J-Link Debugging
- Launch Group

Filter matched 13 of 17 items

Name: PMG1S0_Hello_World Attach (KitProg3_MiniProg4)

Main | Debugger | Startup | Source | Common | SVD Path

Project: PMG1S0_Hello_World

C/C++ Application: ${cy_prj_path}/build/PMG1-CY7110/Debug/mtb-example-pmg1-hello-world.elf

Build (if required) before launching

Build Configuration: Select Automatically

Enable auto build | Disable auto build

[✓ Use workspace settings] | Configure Workspace Settings...

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 and the debug tools appear on the toolbar.

3.4.7.2 Debug through the 10-pin SWD header

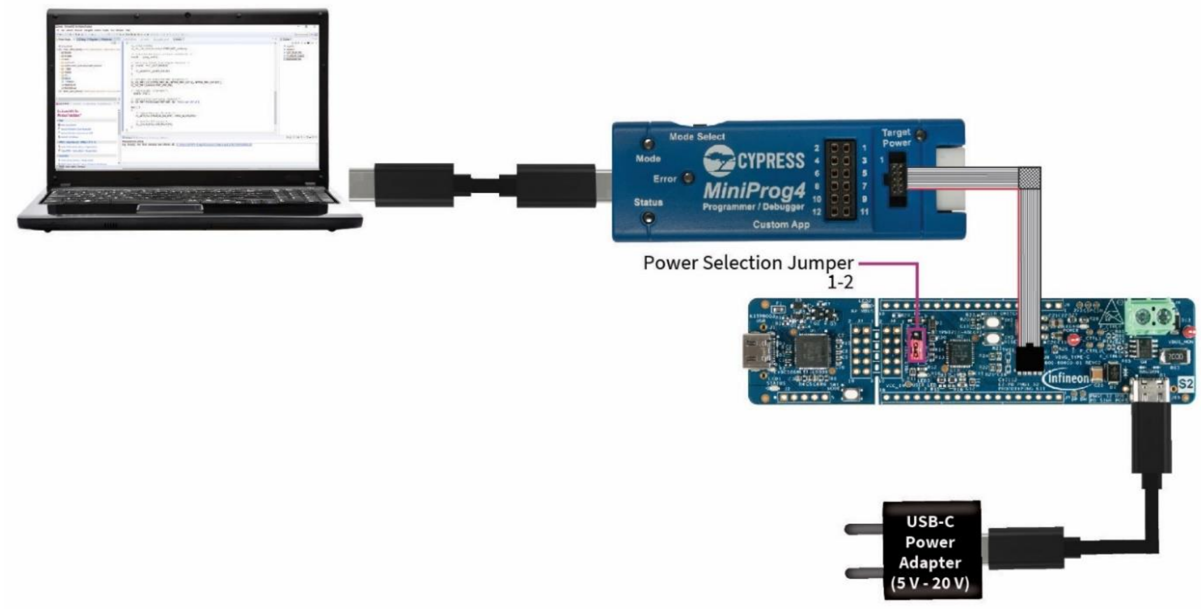


Figure 27 Debugging through the 10-pin SWD header

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 5 to 7 listed in **Debugging through the KitProg3 interface**.

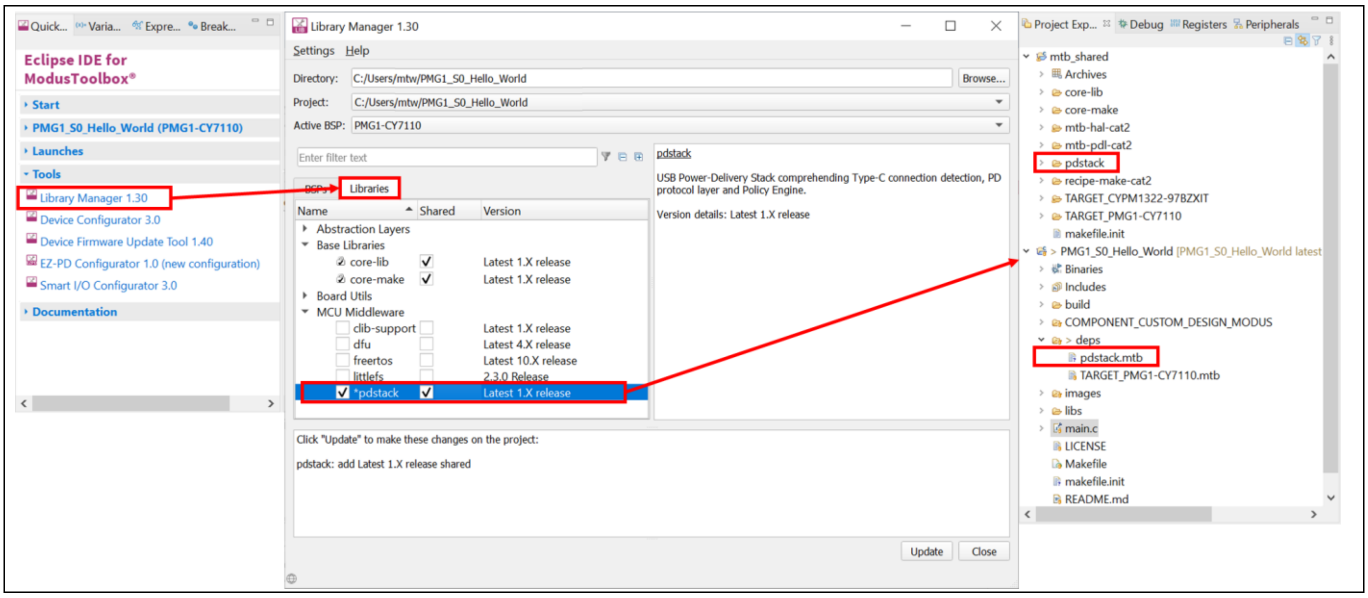
3.5 Add middleware

This section explains the steps to add middleware to your active application using the Library Manager. For the 'Hello World' example project, no additional middleware needs to be added.

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 **Libraries** tab.
3. Under **Board Utils or MCU Middleware**, select and enable the required middleware.
4. Click **Update**.

The files necessary to use the middleware are added in the *mtb_shared* folder as well as in the *.mtb* file under the *deps* folder. **Figure 28** shows an example of adding a middleware. The *.mtb* file pointing to the PDStack middleware is added to the application project. The middleware content is also downloaded and placed inside the corresponding folder called **pdstack.**



Directory: C:/Users/mtw/PMG1_S0_Hello_World

Project: C:/Users/mtw/PMG1_S0_Hello_World

Active BSP: PMG1-CY7110

Name	Shared	Version
Abstraction Layers		
Base Libraries		
core-lib	✓	Latest 1.X release
core-make	✓	Latest 1.X release
Board Utils		
MCU Middleware		
clib-support		Latest 1.X release
dfu		Latest 4.X release
freertos		Latest 10.X release
littlefs		2.3.0 Release
✓ *pdstack	✓	Latest 1.X release

pdstack

USB Power-Delivery Stack comprehending Type-C connection detection, PD protocol layer and Policy Engine.

Version details: Latest 1.X release

Click "Update" to make these changes on the project:

pdstack: add Latest 1.X release shared

Figure 28 **Add PDStack middleware in a ModusToolbox™ software application**

3.6 USB-PD sink example

A code example for USBPD Sink can also be found in the Select Application dialog, select "USBPD_Sink" template application.

The PMG1 MCU devices support a USB-PD 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 is deep sleep state where all clocks are disabled and only limited hardware blocks are enabled, for most of its working time. Interrupts in the USB-PD block are configured to detect any changes that happen while the device is in sleep and wake it up for further processing.

An Over-Voltage (OV) comparator in the USB-PD 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.

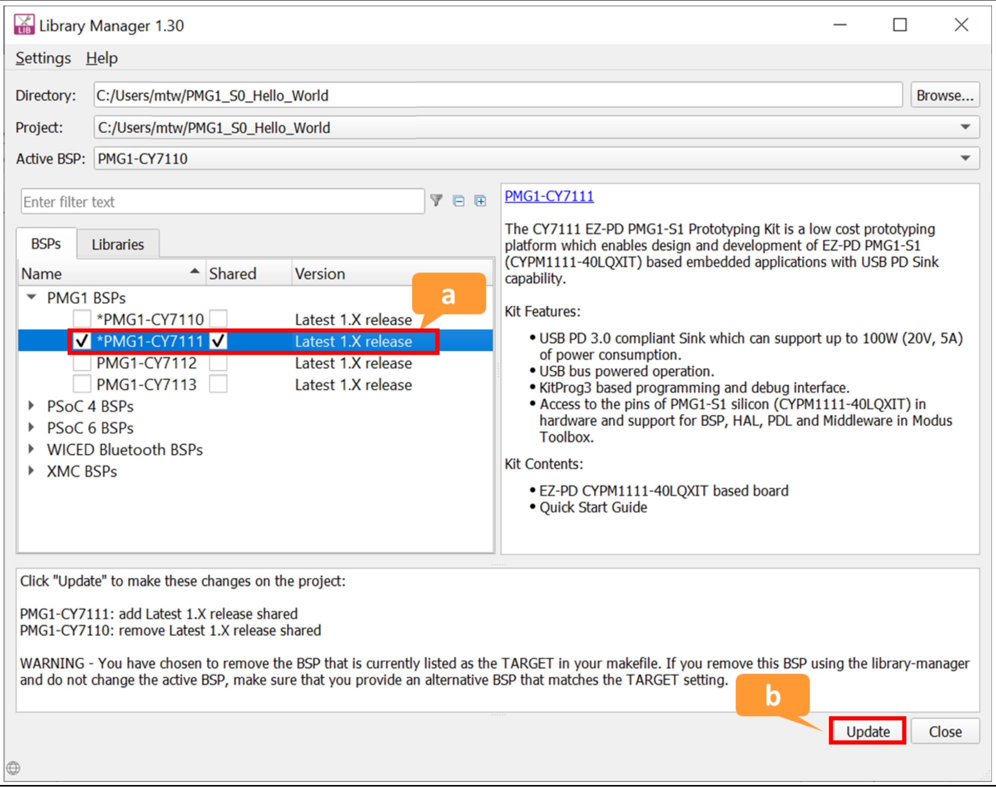
3.7 Porting the project across PMG1 MCUs

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. Please refer below steps for porting the project across PMG1 MCUs.

1. Open the Library Manager from the Quick Link.
2. Do the following in the Library Manager:
   a) Select the new BSP to add it and deselect the current one to remove it. In **Figure 29**, you can see 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**.



Directory: C:/Users/mtw/PMG1_S0_Hello_World
Project: C:/Users/mtw/PMG1_S0_Hello_World
Active BSP: PMG1-CY7110

Name	Shared	Version
PMG1 BSPs		
*PMG1-CY7110		Latest 1.X release
✓ *PMG1-CY7111	✓	Latest 1.X release
PMG1-CY7112		Latest 1.X release
PMG1-CY7113		Latest 1.X release
PSoC 4 BSPs		
PSoC 6 BSPs		
WICED Bluetooth BSPs		
XMC BSPs		

PMG1-CY7111

The CY7111 EZ-PD PMG1-S1 Prototyping Kit is a low cost prototyping platform which enables design and development of EZ-PD PMG1-S1 (CYPM1111-40LQXIT) based embedded applications with USB PD Sink capability.

Kit Features:
- USB PD 3.0 compliant Sink which can support up to 100W (20V, 5A) of power consumption.
- USB bus powered operation.
- KitProg3 based programming and debug interface.
- Access to the pins of PMG1-S1 silicon (CYPM1111-40LQXIT) in hardware and support for BSP, HAL, PDL and Middleware in Modus Toolbox.

Kit Contents:
- EZ-PD CYPM1111-40LQXIT based board
- Quick Start Guide

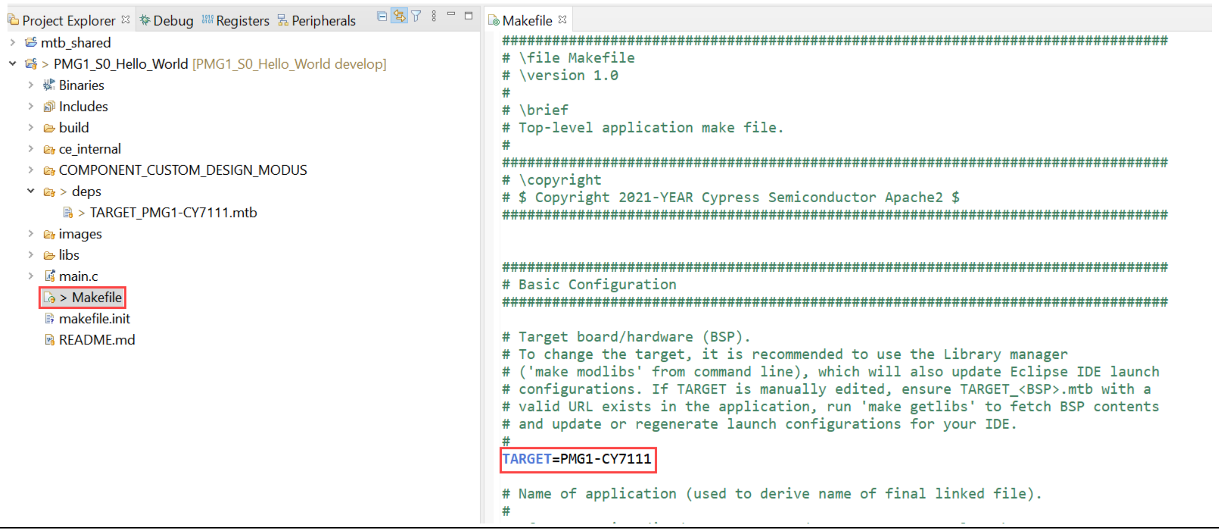
Click "Update" to make these changes on the project:

PMG1-CY7111: add Latest 1.X release shared
PMG1-CY7110: remove Latest 1.X release shared

WARNING - You have chosen to remove the BSP that is currently listed as the TARGET in your makefile. If you remove this BSP using the library-manager and do not change the active BSP, make sure that you provide an alternative BSP that matches the TARGET setting.

Figure 29 Porting a project to a different BSP

3. In the Makefile, change the target name to that of the new BSP selected in previous step. See **Figure 30**.



```
################################################################################
# \file Makefile
# \version 1.0
#
# \brief
# Top-level application make file.
#
################################################################################
# \copyright
# $ Copyright 2021-YEAR Cypress Semiconductor Apache2 $
################################################################################


################################################################################
# Basic Configuration
################################################################################

# Target board/hardware (BSP).
# To change the target, it is recommended to use the Library manager
# ('make modlibs' from command line), which will also update Eclipse IDE launch
# configurations. If TARGET is manually edited, ensure TARGET_<BSP>.mtb with a
# valid URL exists in the application, run 'make getlibs' to fetch BSP contents
# and update or regenerate launch configurations for your IDE.
#
TARGET=PMG1-CY7111

# Name of application (used to derive name of final linked file).
#
```

Figure 30 Edit the target name

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

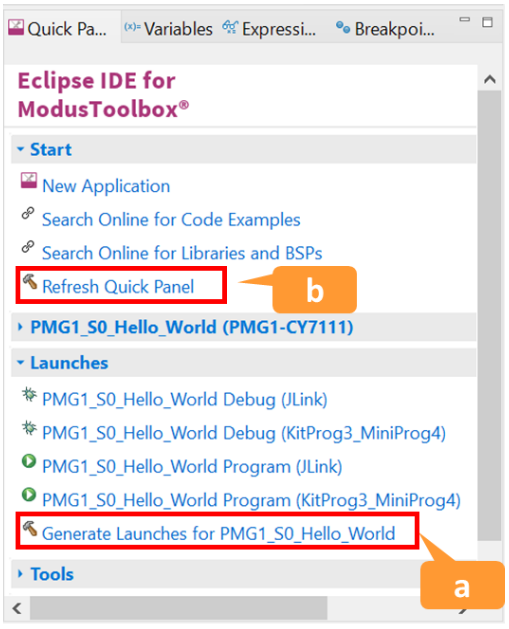
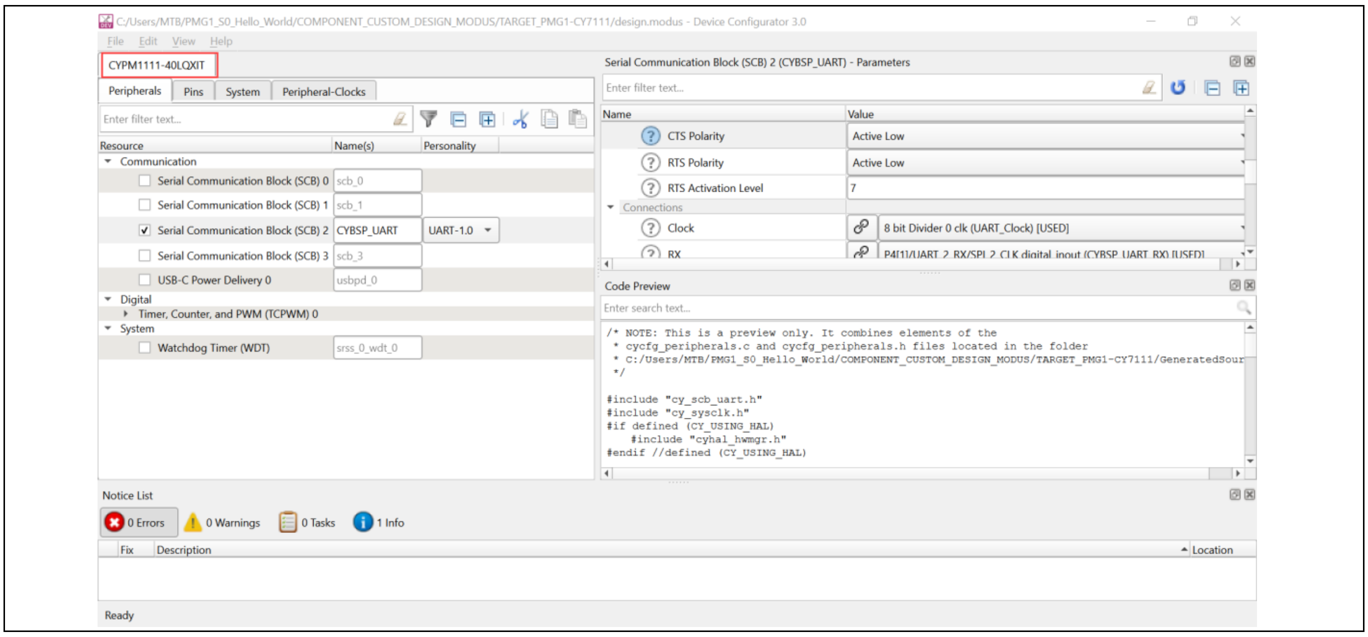

Quick Pa...
Variables
Expressi...
Breakpoi...
Eclipse IDE for ModusToolbox®
Start
New Application
Search Online for Code Examples
Search Online for Libraries and BSPs
Refresh Quick Panel
b
PMG1_S0_Hello_World (PMG1-CY7111)
Launches
PMG1_S0_Hello_World Debug (JLink)
PMG1_S0_Hello_World Debug (KitProg3_MiniProg4)
PMG1_S0_Hello_World Program (JLink)
PMG1_S0_Hello_World Program (KitProg3_MiniProg4)
Generate Launches for PMG1_S0_Hello_World
a
Tools

Figure 31 Regenerate launch configurations

Now, the Device Configurator is updated to the new BSP, as shown in **Figure 32**.



C:/Users/MTB/PMG1_S0_Hello_World/COMPONENT_CUSTOM_DESIGN_MODUS/TARGET_PMG1-CY7111/design.modus - Device Configurator 3.0

CYPM1111-40LQXIT

Resource	Name(s)	Personality
Communication		
☐ Serial Communication Block (SCB) 0	scb_0	
☐ Serial Communication Block (SCB) 1	scb_1	
✓ Serial Communication Block (SCB) 2	CYBSP_UART	UART-1.0
☐ Serial Communication Block (SCB) 3	scb_3	
☐ USB-C Power Delivery 0	usbpd_0	
Digital		
Timer, Counter, and PWM (TCPWM) 0		
System		
☐ Watchdog Timer (WDT)	srss_0_wdt_0	

Serial Communication Block (SCB) 2 (CYBSP_UART) - Parameters

Name	Value
CTS Polarity	Active Low
RTS Polarity	Active Low
RTS Activation Level	7
Connections	
Clock	8 bit Divider 0 clk (UART_Clock) [USED]
RX	P4[1]/UART_2_RX/SPI_2_CLK digital_inout (CYBSP_UART_RX) [USED]

Code Preview

```
/* NOTE: This is a preview only. It combines elements of the
 * cycfg_peripherals.c and cycfg_peripherals.h files located in the folder
 * C:/Users/MTB/PMG1_S0_Hello_World/COMPONENT_CUSTOM_DESIGN_MODUS/TARGET_PMG1-CY7111/GeneratedSour
 */

#include "cy_scb_uart.h"
#include "cy_sysclk.h"
#if defined (CY_USING_HAL)
    #include "cyhal_hwmgr.h"
#endif //defined (CY_USING_HAL)
```

Notice List

0 Errors 0 Warnings 0 Tasks 1 Info

Fix	Description	Location

Ready

Figure 32 Device configurator with new BSP

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

4 Summary

This application note explored the PMG1 MCU device architecture and the associated development tools. PMG1 MCU is a programmable embedded system-on-chip with configurable analog and digital peripheral functions, memory, and CM0/CM0+ CPU system.

Related documents

For a complete and updated list of PMG1 MCU code examples, visit our **GitHub** repository. For more PMG1 MCU-related documents, visit the **PMG1 MCU product web page**.

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	Over Current Protection
OVP	Over Voltage Protection
OVT	Over Voltage 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.

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

MiniProg3 / **MiniProg4**: Programming hardware for development that is used to program PMG1 MCU devices on your custom board or PMG1 MCU kits that do not support a built-in programmer.

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 provide 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 the lowest power, 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 version	Date of release	Description of changes
**	2021-02-26	Initial release
*A	2021-07-28	Added **1.5 PMG1-S3 MCU** section Updated **Table 2** and **Figure 12** Added a note in **3.4.5** Added **3.4.7 Part 7: Debug the application** section
*B	2022-02-10	Updated section **3.4.6 Part 6: Test your design** to describe the UART connection step on older revisions of the kit board

Edition 2022-02-10

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about this document?

Go to www.infineon.com/support

Document reference

002-32553 Rev. *B