

Getting started with PSoC™ 4

About this document

Scope and purpose

This application note introduces you to PSoC™ 4, an Arm® Cortex®-M0/M0+ based programmable system-on-chip. It helps you explore the PSoC™ 4 architecture and development tools and shows you how to create your first project using PSoC™ Creator and ModusToolbox™, the development tools for PSoC™ 4. This application note also guides you to more resources to accelerate in-depth learning about PSoC™ 4.

Intended audience

This application note is intended for engineers new to PSoC™ and ModusToolbox™, and those with experience in working with embedded microcontrollers.

Associated part family

All **PSoC™ 4** parts

Software version

PSoC™ Creator 4.3 SP2 or higher, **ModusToolbox™** 2.3 or higher

More code examples? We heard you.

To access an ever-growing list of hundreds of PSoC™ code examples, please visit our [code examples web page](#). You can also explore the video training library [here](#).

Table of contents

About this document	1
Table of contents	1
1 Introduction	3
2 PSoC™ resources	4
2.1 Firmware/application development	4
2.2 Choosing an IDE.....	6
2.2.1 Why choose ModusToolbox™?.....	7
2.3 ModusToolbox™ resources	7
2.3.1 PSoC™ 4 software resources	7
2.3.2 Configurators.....	8
2.3.3 Software development for PSoC™ 4.....	8
2.3.4 Code example.....	10
2.3.5 ModusToolbox™ help	11
2.4 PSoC™ Creator	12
2.4.1 Code examples.....	12
2.4.2 PSoC™ Creator help.....	14
2.5 Technical support.....	14
3 PSoC™ 4 feature set	15
4 PSoC™ is more than an MCU	23

Table of contents

4.1	The concept of PSoC™ Creator Components	24
5	My first PSoC™ 4 design using ModusToolbox™	25
5.1	Before you begin	25
5.1.1	Have you installed ModusToolbox™?	25
5.1.2	Do you have a development kit or prototyping kit?	25
5.2	Using these instructions.....	25
5.3	About the design	26
5.4	Part 1: Creating a new application.....	26
5.4.1	Select a new workspace.....	26
5.4.2	Create a new ModusToolbox™ application	26
5.4.3	Select a target PSoC™ 4 development kit	27
5.5	Part 2: Viewing and modifying the design	29
5.5.1	Project structure	29
5.5.2	Modify the design	32
5.6	Part 3: Writing firmware	37
5.6.1	Firmware flow.....	38
5.7	Part 4: Building the application	40
5.7.1	Build the application.....	40
5.8	Part 5: Programming the device	41
5.8.1	Program the application.....	42
5.9	Part 6: Testing your design.....	43
5.9.1	Select the serial port	43
5.9.2	Set the baud rate.....	44
5.9.3	Reset the device	44
6	My first PSoC™ 4 design using PSoC™ Creator	45
6.1	Before you begin	45
6.1.1	Have you installed PSoC™ Creator?.....	45
6.1.2	Do you have a development kit or prototyping kit?	45
6.1.3	Want to see the project in action?	45
6.2	About the design	45
6.3	Part 1: Create the design	46
6.4	Part 2: Program the device.....	55
7	Summary	57
	References.....	58
	Revision history.....	59

Introduction

1 Introduction

PSoC™ 4 is a true programmable embedded system-on-chip, integrating custom analog and digital peripheral functions, memory, and an Arm® Cortex®-M0 or Cortex®-M0+ microcontroller on a single chip. This type of system is different from most mixed-signal embedded systems, which use a combination of a microcontroller unit (MCU) and external analog and digital peripherals. Such systems typically require many integrated circuits in addition to the MCU, such as opamps, ADCs, and Application-specific Integrated Circuit (ASICs).

PSoC™ 4 provides a low-cost alternative to the combination of MCU and external ICs. In addition to reducing overall system cost, the programmable analog and digital subsystems allow great flexibility, in-field tuning of the design, and speedy time to market.

The capacitive touch-sensing feature in PSoC™ 4, known as CAPSENSE™, offers unprecedented signal-to-noise ratio; best-in-class waterproofing; and a wide variety of sensor types such as buttons, sliders, trackpads, and proximity sensors. PSoC™ 4 offers a best-in-class current consumption of 150 nA while retaining SRAM, programmable logic, and the ability to wake up from an interrupt. PSoC™ 4 consumes only 20 nA while maintaining wakeup capability in its non-retention power mode. The PSoC™ 4 family of devices also contain PSoC™ 4 Bluetooth® LE, which integrates a Bluetooth® Low Energy radio system. For more details on PSoC™ 4 Bluetooth® LE, see [AN91267](#).

Using this document

The next few pages describe PSoC™ 4 and the advantages of designing with PSoC™, ModusToolbox™, and PSoC™ Creator. Or, you can jump right in and quickly build a simple design in ModusToolbox™ – go to [My first PSoC™ 4 design using ModusToolbox™](#) in PSoC™ Creator – go to [My first PSoC™ 4 design using PSoC™ Creator](#).

PSoC™ resources

2 PSoC™ resources

The wealth of data available [here](#) can help you select the right PSoC™ device, and quickly and effectively integrate the device into your design. The following is an abbreviated list for PSoC™ 4:

- **Overview:** [PSoC™ portfolio](#), [PSoC™ roadmap](#)
- **Product selectors:** [PSoC™ 4](#). In addition, [PSoC™ Creator](#) includes a device selection tool.
- **Datasheets** describe and provide electrical specifications for each family.
- **Application notes** cover a broad range of topics, from basic to advanced level, and include the following:
 - [AN88619](#): PSoC™ 4 hardware design considerations
 - [AN73854](#): Introduction to bootloaders
 - [AN89610](#): Arm® Cortex® code optimization
 - [AN86233](#): PSoC™ 4 low-power modes and power reduction techniques
 - [AN57821](#): Mixed-signal circuit board layout
 - [AN89056](#): PSoC™ 4 - IEC 60730 class B and IEC 61508 SIL Safety Software Library
 - [AN64846](#): Getting started with CAPSENSE™
 - [AN85951](#): PSoC™ 4 and PSoC™ 6 MCU CAPSENSE™ design guide
- **Code examples** demonstrate product features and usage – see PSoC™ Creator [Code examples](#) and ModusToolbox™ [Code example](#).
- **Technical reference manuals (TRMs):** Provide detailed descriptions of the architecture and registers in each PSoC™ 4 device family.
- **PSoC™ 4 programming specification** provides the information necessary to program PSoC™ 4 nonvolatile memory.
- **Development tools:**
 - [CY8CKIT-040](#), [CY8CKIT-041](#), [CY8CKIT-042](#), [CY8CKIT-044](#), [CY8CKIT-046](#), [CY8CKIT-042-BLE](#), [CY8CKIT-045S](#), and [CY8CKIT-041S-MAX](#) PSoC™ 4 pioneer kits are easy-to-use and inexpensive development platforms. These include connectors for Arduino-compatible shields and Digilent® Pmod™ daughter cards.
 - [CY8CKIT-043](#), [CY8CKIT-145](#), [CY8CKIT-147](#) and [CY8CKIT-149](#) are very low-cost prototyping platforms for sampling PSoC™ 4 devices.
 - The [MiniProg3](#) or [MiniProg4](#) kit provides an interface for flash programming and debug.
 - Integrated Development Environment (IDE): There are two development platforms that you can use for application development with PSoC™ 4 – [ModusToolbox™](#) and [PSoC™ Creator](#)
 - [PSoC™ 4 CAD libraries](#) provide footprint and schematic support for common tools. [IBIS models](#) are also available.
- **Training videos** are available on a wide range of topics including the [PSoC™ 4 101 series](#)
- **Cypress developer community** enables connection with fellow PSoC™ developers around the world, 24 hours a day, 7 days a week, and hosts a dedicated [PSoC™ 4 MCU](#) community

2.1 Firmware/application development

There are two development platforms that you can use for application development with PSoC™ 4:

- **ModusToolbox™:** ModusToolbox™ software includes configuration tools, low-level drivers, middleware libraries, and operating system support, as well as other packages that enable you to create MCU and wireless applications. It also includes the optional Eclipse IDE.

PSoC™ resources

ModusToolbox™ supports stand-alone device and middleware configurators that can be launched from the Eclipse IDE. Use the configurators to set the configuration of different blocks in the device and generate code that can be used in firmware development. ModusToolbox™ supports all PSoC™ 6 MCU and the latest PSoC™ 4 MCU devices. **Table 1** lists the supported PSoC™ 4 devices. It is recommended that you use ModusToolbox™ for all application development for supported PSoC™ 4 devices. See **ModusToolbox™ resources** for more information.

Table 1 List of PSoC™ 4 devices supported in ModusToolbox™

Devices ¹	ModusToolbox™	PSoC™ Creator
PSoC™ 4000S, PSoC™ 4100S, PSoC™ 4100S Plus, PSoC™ 4100S Plus 256K	Yes	Yes
PSoC™ 4100S Max	Yes	No
All other PSoC™ 4 devices	No	Yes

The libraries and enablement software are available on [GitHub](#).

ModusToolbox™ tools and resources can also be used in the command line. See **Running ModusToolbox™ from the command line** for detailed documentation.

- PSoC™ Creator: PSoC™ Creator** is a free Windows-based IDE. It enables concurrent hardware and firmware design of PSoC™ 3, PSoC™ 4, PSoC™ 5LP, and PSoC™ 6 MCU systems. Applications are created using schematic capture and over 150 pre-verified, production-ready peripheral Components.

¹ See **PSoC™ 4 feature set** for complete PSoC™ 4 portfolio.

2.2 Choosing an IDE

Figure 1 helps you choose an appropriate IDE.

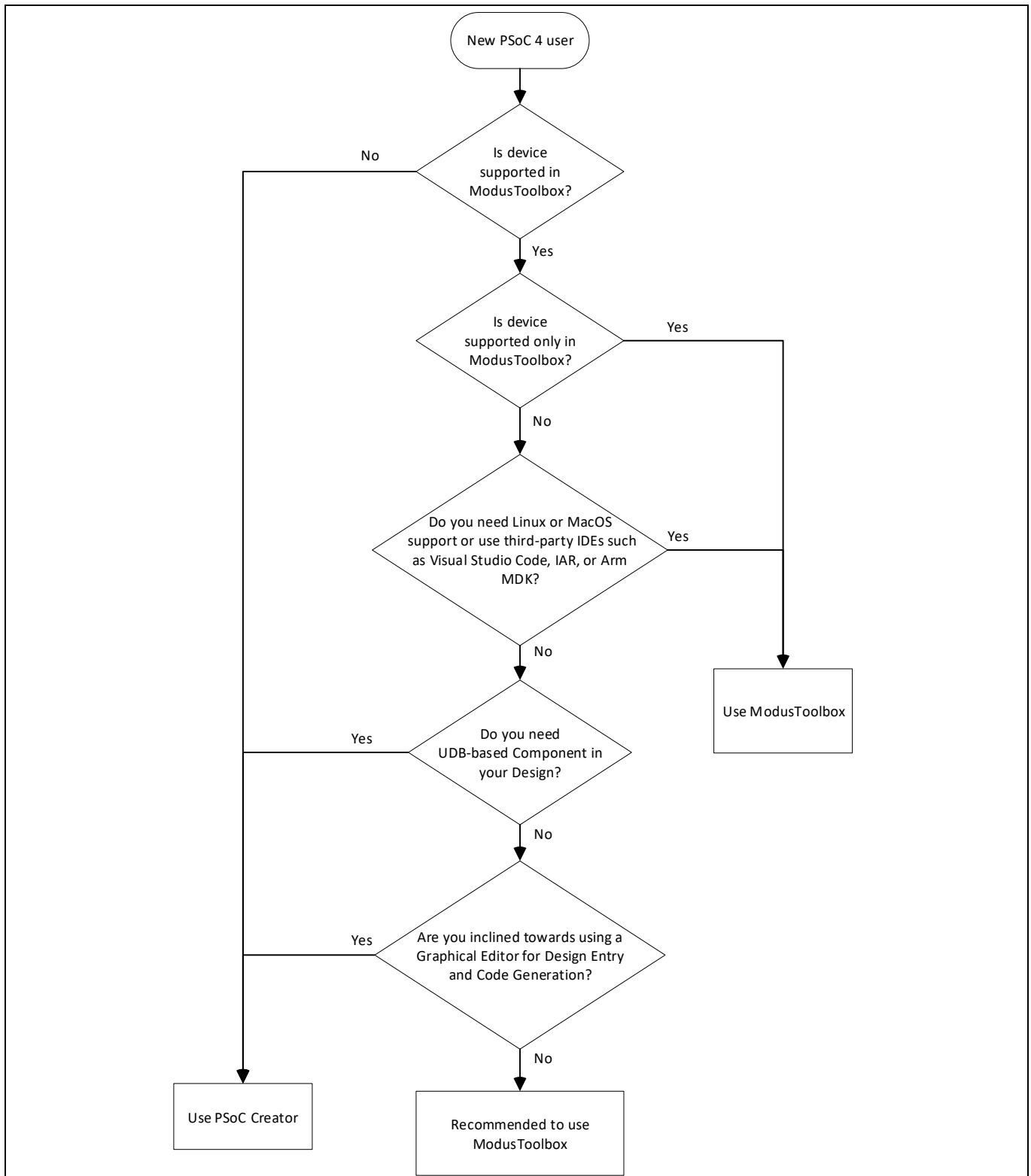


Figure 1 Choosing an IDE

PSoC™ resources

2.2.1 Why choose ModusToolbox™?

- Comprehensive: It has the resources you need.
- Flexible: You can use the resources in your own workflow.
- Atomic: You can get just the resources you want.
- A large collection of code repositories on [GitHub](#), including:
 - Board Support Packages (BSPs) aligned with Infineon kits
 - Low-level resources, including a Hardware Abstraction Layer (HAL) and Peripheral Driver Library (PDL)
 - Middleware enabling industry-leading features such as CAPSENSE™
 - An extensive set of thoroughly tested code example applications
- Supported across Windows, Linux, and macOS platforms.
- IDE-neutral and supports third-party IDEs, including Visual Studio Code, Arm® MDK (µVision), and IAR Embedded Workbench.
- Choose ModusToolbox™ if you have prior experience with Eclipse-based tools and want to take advantage of the power and extensibility of the optional Eclipse-based IDE for ModusToolbox™.

2.3 ModusToolbox™ resources

ModusToolbox™ is a set of tools and software that enables an immersive development experience for creating converged MCU and wireless systems, and enables you to integrate Infineon devices into your existing development methodology.

Eclipse IDE for ModusToolbox™ is a multi-platform development environment that supports application configuration and development. ModusToolbox™ installer includes the design configurators and tools, and the build system infrastructure. The build system infrastructure includes the new project creation wizard that can be run independent of the Eclipse IDE, the make infrastructure, and other tools.

2.3.1 PSoC™ 4 software resources

ModusToolbox™ also includes Infineon-provided software resources delivered via [GitHub](#). These include:

- BSP – BSP is the layer of firmware containing board-specific drivers and other functions. The BSP 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 the PDL library for PSoC™ 4, and has macros for board peripherals. Custom BSPs can be created to enable support for end-application boards. See the [ModusToolbox™ Library Manager user guide](#) for more information.
- HAL – The HAL provides a high-level interface to configure and use hardware blocks on Infineon MCUs. It is a generic interface that can be used across multiple product families. The focus on ease-of-use and portability means that the HAL does not expose all low-level peripheral functionalities. HAL wraps the low-level drivers (like PSoC™ 4 PDL) and provides a high-level interface to the MCU. The interface is abstracted to work on any MCU. This helps you write application firmware independent of the target MCU.
- The HAL can be combined with platform-specific libraries (such as PSoC™ 4 PDL) within a single application. You can leverage the HAL's simpler and more generic interface for most of an application, even if one portion requires finer-grained control.
- PSoC™ 4 PDL – The PDL integrates device header files, startup code, and peripheral drivers into a single package. PDL supports the PSoC™ 4 device family. The drivers abstract the hardware functions into a set of easy-to-use APIs. These are fully documented in the PDL API Reference.

PSoC™ resources

- PDL reduces the need to understand register usage and bit structures, thus easing software development for the extensive set of peripherals in the PSoC™ 4 series. You configure the driver for your application, and then use API calls to initialize and use the peripheral.
- Extensive middleware libraries that provides specific capabilities to an application. The [available middleware](#) spans across connectivity (Bluetooth®, AWS IoT, Bluetooth® LE, Secure Sockets) to PSoC™-specific functionality (CAPSENSE™). The middleware is delivered as libraries via [GitHub](#) repositories.
- Utilities, Makefiles, scripts, and other configuration software.

2.3.2 Configurators

ModusToolbox™ provides graphical applications called Configurators that make it easier to configure a hardware block. For example, instead of having to search through the documentation to configure a serial communication block (SCB) as a UART with a desired configuration, open the appropriate 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 a stand-alone, in conjunction with other tools, or launched from the Eclipse 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 PSoC™ 4 applications, the available Configurators include:

- **Device Configurator:** Sets up the system (platform) functions and the basic peripherals (for example, UART, Timer, PWM).
- **CAPSENSE™ Configurator and Tuner:** Configures CAPSENSE™ and generates the required code.
- **Smart I/O Configurator:** Configures the Smart I/O.

Each of these Configurators create their own files (for example, *design.cycapsense* for CAPSENSE™). The configurator files (*design.modus* or *design.cycapsense*) are usually provided with the BSP. 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 files provided by the BSP. See [ModusToolbox™ help](#) for more details.

2.3.3 Software development for PSoC™ 4

Significant source code and tools are provided to enable software development for PSoC™ 4 devices. You can use the tools to specify how you want to configure the hardware, generate code that you can use in your firmware, and include various middleware libraries, such as CAPSENSE™, for additional functionality. This source code makes it easier to develop firmware for supported devices. It helps you to quickly customize and build firmware without the need to understand the register set.

In the ModusToolbox™ environment, you can use configurators to configure either the device or a middleware library, like the CAPSENSE™ functionality.

The driver code is delivered as the *mtb-pdl-cat2* library. Middleware is delivered as separate libraries for each feature/function.

If you are working at the register-level, regardless of whether you use Eclipse IDE, a third-party IDE, or the command line, see the driver source code from the PDL. PDL includes all device-specific header files and

PSoC™ resources

startup code you need for your project. It also serves as a reference for each driver. PDL is provided as source code, so you can see how it accesses the hardware at the register level.

Some devices do not support particular peripherals. PDL is a superset of all drivers for any supported device. This superset design means that:

- All API elements needed to initialize, configure, and use a peripheral are available.
- PDL is useful across various PSoC™ 4 devices, regardless of available peripherals.
- PDL includes error checking to make sure that the targeted peripheral is present on the selected device.

This enables the code to maintain compatibility across the PSoC™ 4 device family as long as the peripherals are available. A device header file specifies the peripherals that are available for a device. If your code attempts to use an unsupported peripheral, you will get an error during compilation. Before writing the code, see the device datasheet to check if the peripheral is supported.

Figure 2 shows that with the Eclipse IDE for ModusToolbox™ you can:

1. Use Project Creator (**File > New > ModusToolbox Application**) to create a new application for your BSP (kit).
2. Configure peripherals and middleware libraries using configuration tools.
3. Add, update, or remove libraries and BSPs easily and quickly using the Library Manager.
4. Optionally, develop code in an Eclipse-based IDE.

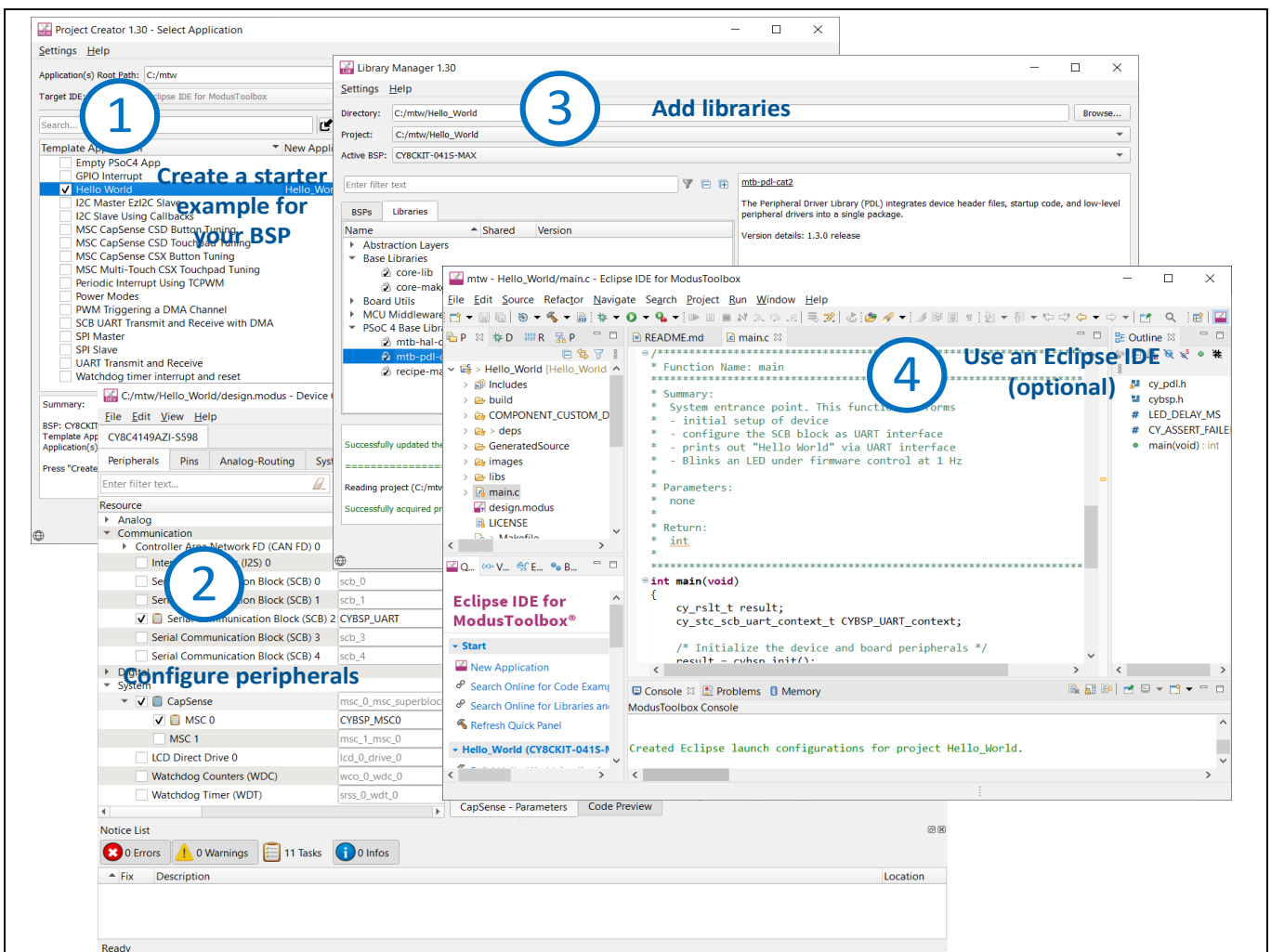


Figure 2 Eclipse IDE for ModusToolbox™ resources and middleware

PSoC™ resources

2.3.4 Code example

The Project Creator tool is used with the ModusToolbox™ software to create applications based on BSPs and code examples. Project Creator is provided as a graphical user interface (GUI) and a command-line interface (CLI) to create applications, which can be used in any software environment

1. Open the Project Creator tool.

Follow the menu **File > New > ModusToolbox Application**.

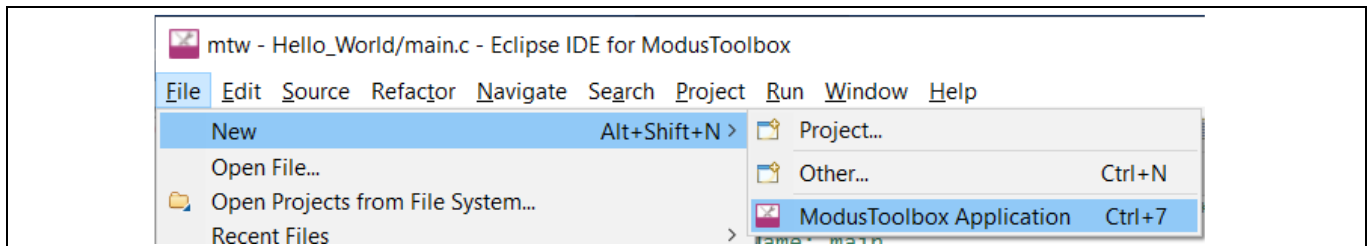


Figure 3 Opening project creator tool

2. Select a target PSoC™ 4 development kit.

In the Choose Board Support Package (BSP) dialog, select your kit from the list.

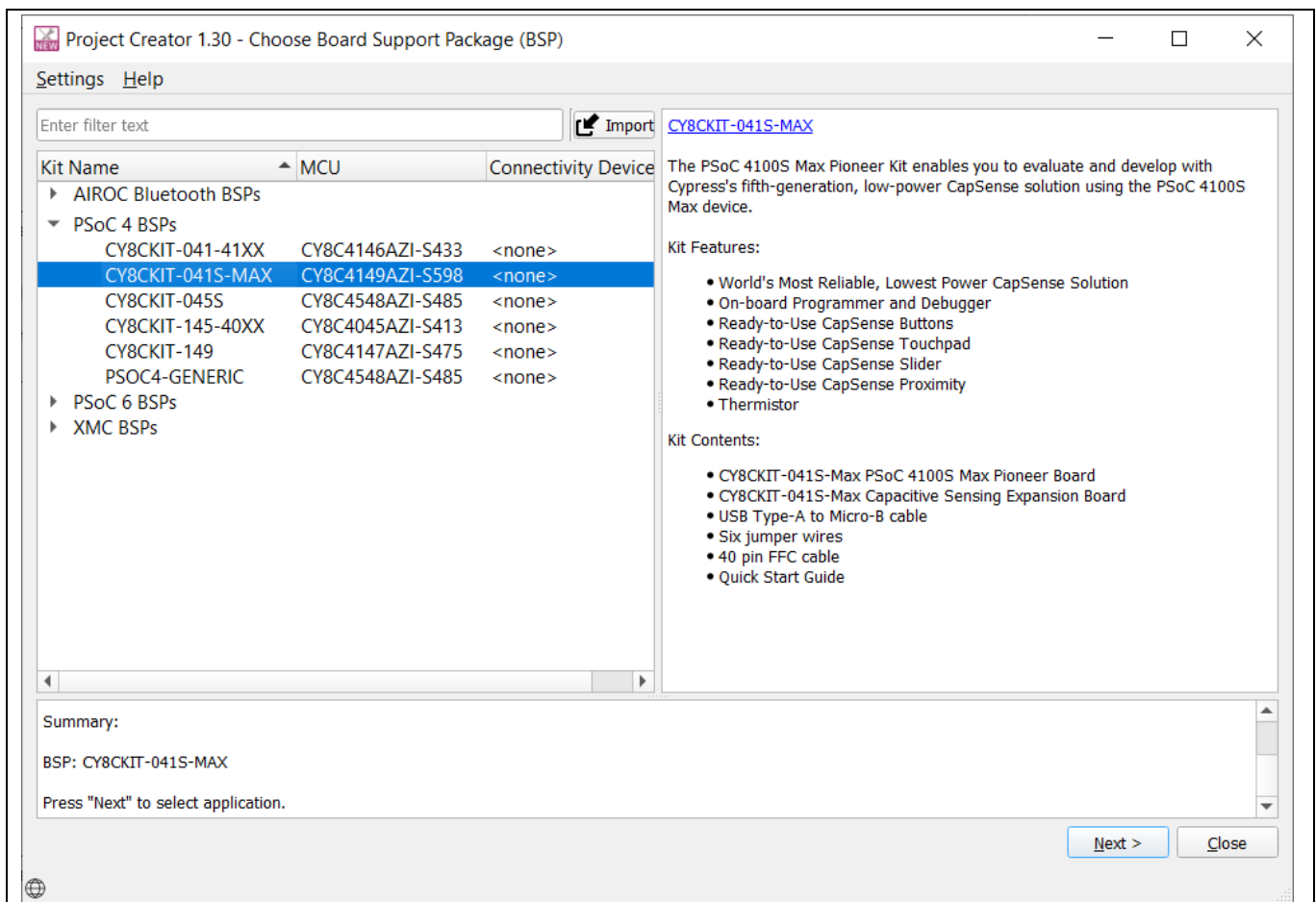


Figure 4 Choosing target hardware

PSoC™ resources

3. Select an application.

The Select **Application** dialog lists the Application applicable for the selected BSP. Select the required application and click **Create**. The selected application will be created.

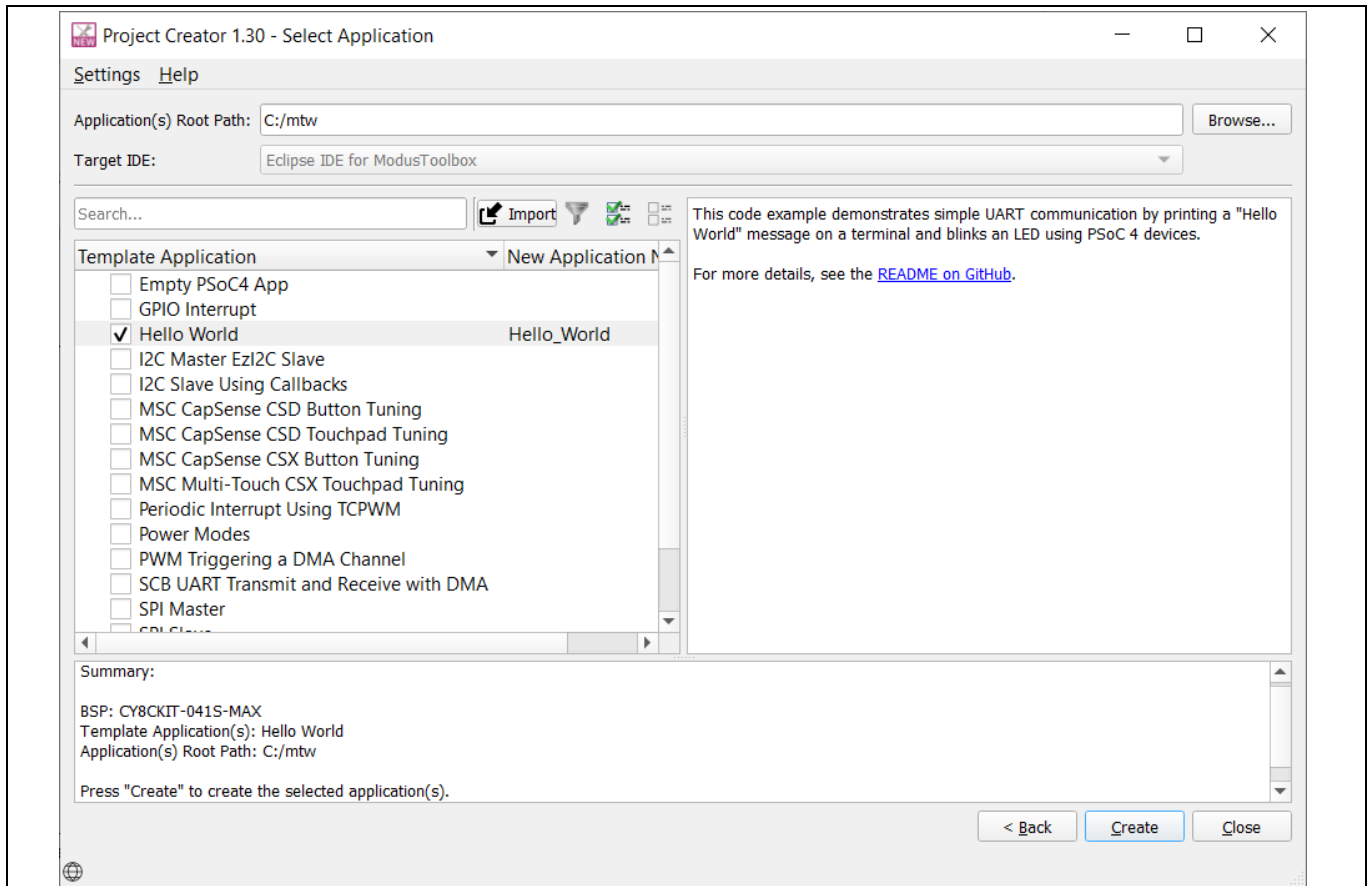


Figure 5 Selecting an application

2.3.5 ModusToolbox™ help

[ModusToolbox™ user guide](#) provides a high-level overview of the ModusToolbox™ software.

Download the latest version of [ModusToolbox™](#) and install it. Launch Eclipse IDE for ModusToolbox™ and navigate to **Help > ModusToolbox General Documentation**:

- **User Guide:** Covers the aspects of building, programming and debugging applications. It also covers various aspects of the tools installed along with the IDE
- **ModusToolbox Documentation Index:** Provides brief descriptions and links to various documentation included in the ModusToolbox™ software.

Release Notes: Describes features for the corresponding release of ModusToolbox™, and lists the known issues, workarounds, and design impacts that you should be aware of. For documentation on Eclipse IDE for ModusToolbox™, navigate to **Help > Eclipse IDE for ModusToolbox Documentation**:

- **Quick Start Guide:** Provides the basics for using Eclipse IDE for ModusToolbox™.
- **User Guide:** Describes how to create applications, and build, program, and debug them using Eclipse IDE
- **Eclipse IDE Survival Guide:** Provides answers to the common issues that might occur while using an Eclipse IDE. Most questions are related to Eclipse IDE and not ModusToolbox™.

PSoC™ resources

2.4 PSoC™ Creator

PSoC™ Creator is an IDE that enables concurrent hardware and firmware editing, compiling and debugging of PSoC™ systems. As **Figure 6** shows, with PSoC™ Creator, you can:

1. Drag and drop Components to build your hardware system design.
2. Co-design your application firmware with the PSoC™ hardware.
3. Configure Components with config tools.
4. Explore the library of more than 100 Components.
5. Review Component datasheets.

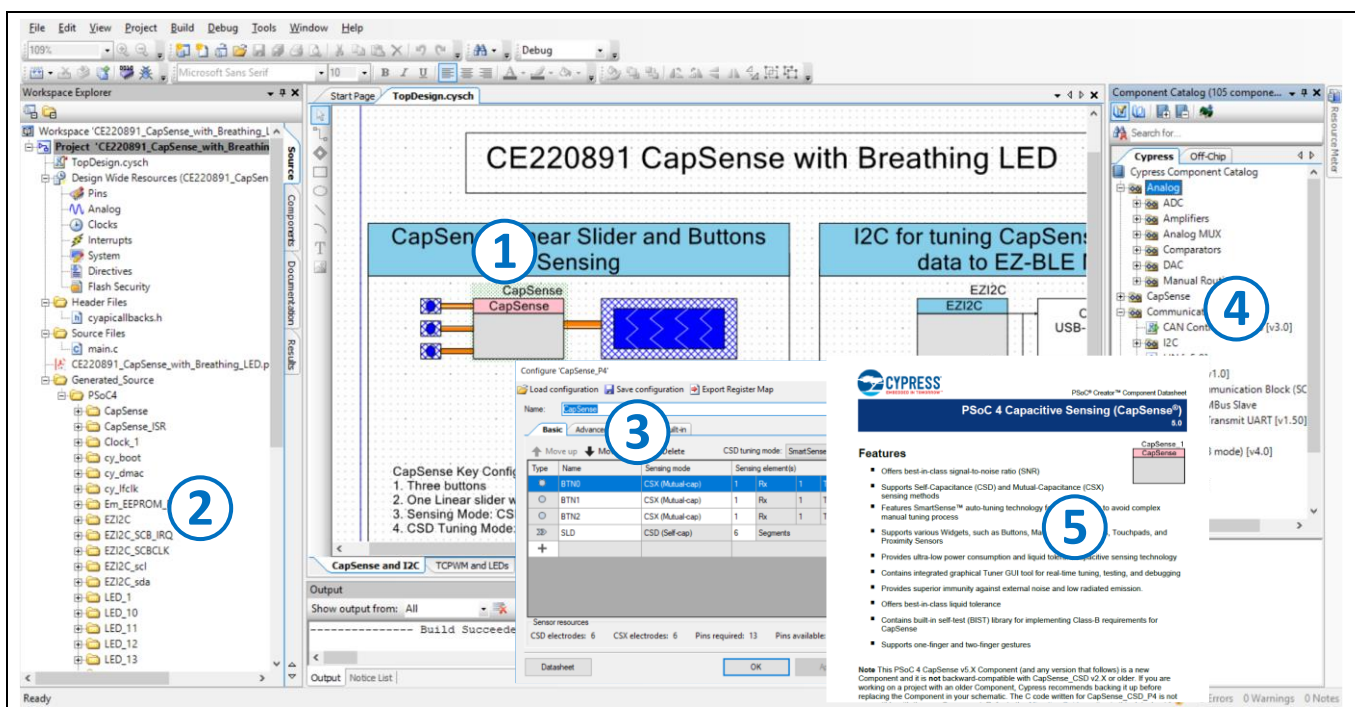


Figure 6 Features of PSoC™ Creator features

2.4.1 Code examples

PSoC™ Creator includes a large number of code example projects. These projects are available from the PSoC™ Creator Start Page, as shown in **Figure 7**.

PSoC™ resources

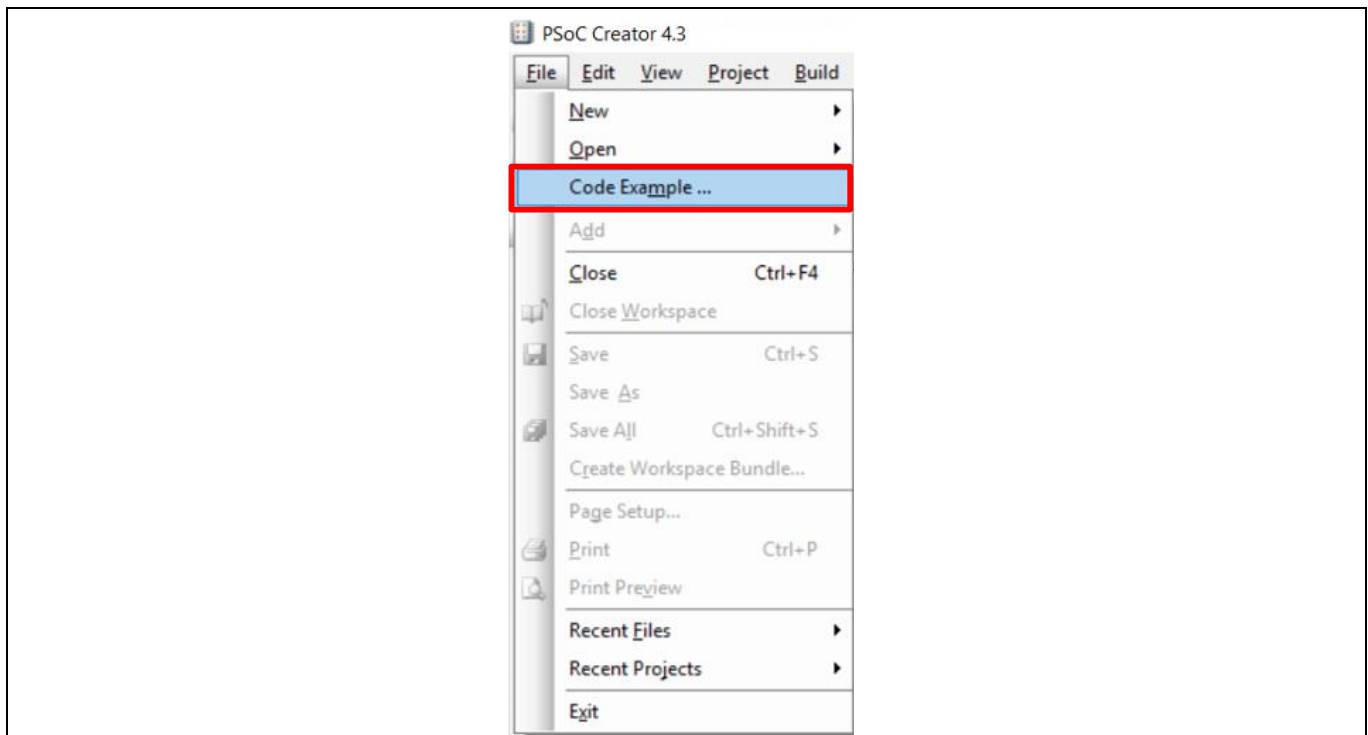


Figure 7 Code examples in PSoC™ Creator

Example projects can speed up your design process by starting you off with a complete design, instead of a blank page. The example projects also show how PSoC™ Creator Components can be used for various applications. Code examples and datasheets are included, as shown in [Figure 8](#).

In the Find Example Project dialog, shown in [Figure 8](#), you can:

- a) Filter examples based on architecture or device family, that is, PSoC™ 3, PSoC™ 4, PSoC™ 5LP or PSoC™ 6 MCU, category, or keyword.
- b) Select from the filtered list of examples.
- c) Review the datasheet for the selection (in the **Documentation** tab)
- d) Review the code example for the selection. You can copy the code from this window and paste in your project, which can help speed up code development. Alternatively, you can create a new project (and a new workspace if needed) based on the selection. This can speed up your design process by starting you off with a complete, basic design. You can then adapt that design to your application.

PSoC™ resources

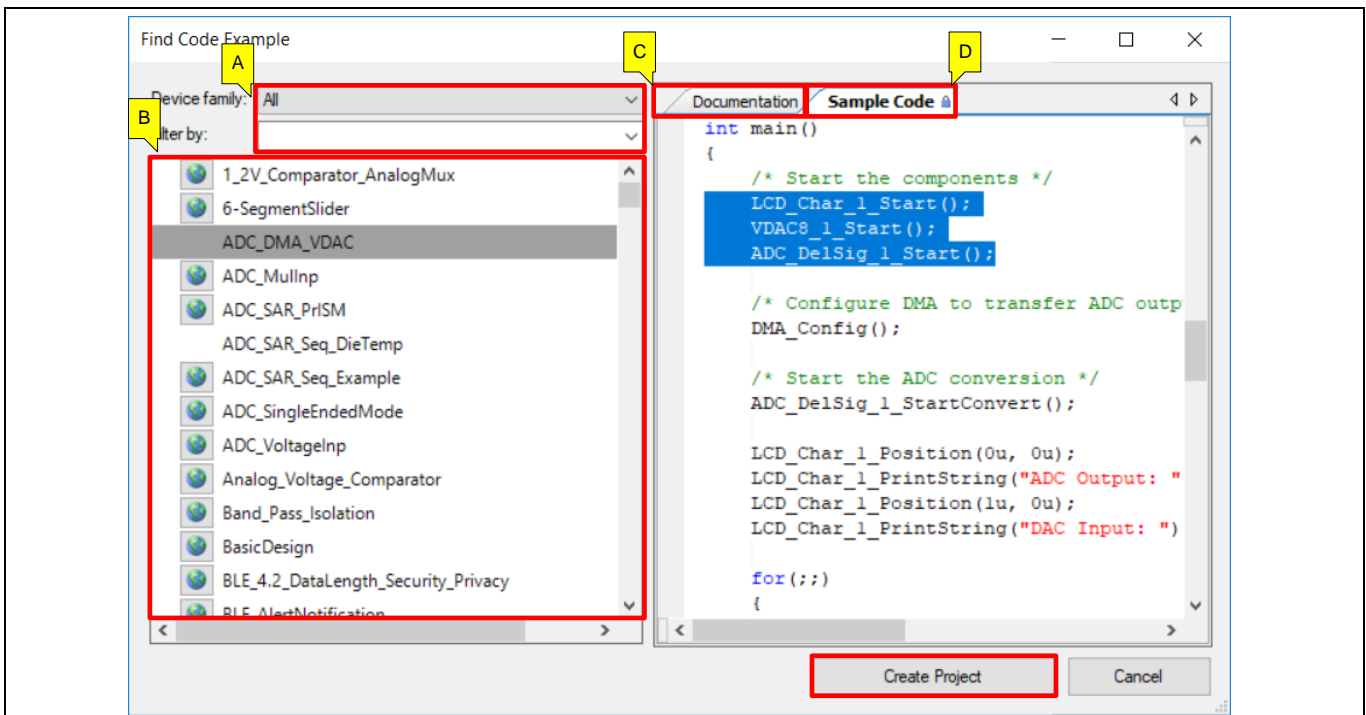


Figure 8 Code example projects with sample code

2.4.2 PSoC™ Creator help

Visit the [PSoC™ Creator home page](#) to download the latest version of PSoC™ Creator. Then, launch PSoC™ Creator and navigate to the following items:

- **Quick Start Guide:** Choose **Help > Documentation > Quick Start Guide**. This guide gives you the basics for developing PSoC™ Creator projects.
- **Simple Component example projects:** Choose **File > Open > Example projects**. These example projects demonstrate how to configure and use PSoC™ Creator Components.
- **System Reference Guide:** Choose **Help > System Reference > System Reference Guide**. This guide lists and describes the system functions provided by PSoC™ Creator.
- **Component datasheets:** Right-click a Component and select **Open Datasheet**. Visit the [PSoC™ 4 Component datasheets](#) page for a list of all PSoC™ 4 Component datasheets.
- **PSoC™ Creator training videos:** These videos provide step-by-step instructions on how to get started with PSoC™ Creator.
- **Document Manager:** PSoC™ Creator provides a document manager to help you to easily find and review document resources. To open the document manager, choose the menu item **Help > Document Manager**.

2.5 Technical support

If you have any questions, our technical support team is happy to assist you. You can create a support request on the [Technical support](#) page. The support team monitors and responds to your questions, issues, and bug reports posted on the [GitHub](#) repositories.

You can also use the following support resources if you need quick assistance.

- [Self-help](#)
- [Local sales office locations](#)

PSoC™ 4 feature set

3 PSoC™ 4 feature set

PSoC™ 4 has an extensive set of features, which include a CPU and memory subsystem, a digital subsystem, an analog subsystem, and system resources, as shown in Figure 9. The following sections give brief descriptions of each feature. For more information, see the PSoC™ 4 family device datasheets, technical reference manuals (TRMs), and application notes listed in PSoC™ resources.

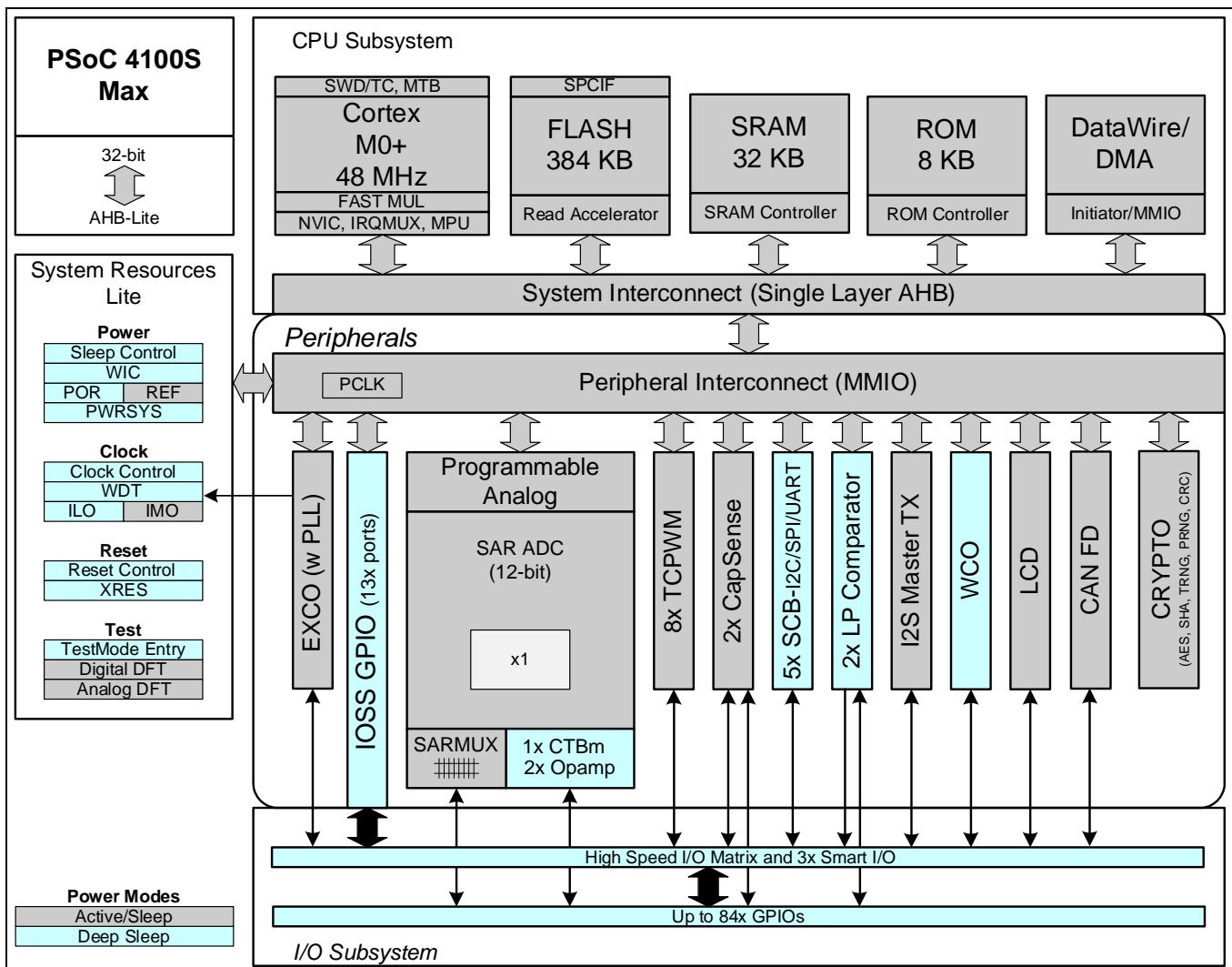


Figure 9 PSoC™ 4 architecture (PSoC™ 4100S Max)

The PSoC™ 4 portfolio consists of several families of Arm® CM0 and CM0+ microcontrollers. Most devices in the portfolio has CAPSENSE™ technology for capacitive-sensing applications. Other key features of the PSoC™ 4 portfolio include a customizable analog front end through programmable analog blocks and wired and wireless connectivity options such as USB, Controller Area Network (CAN), and Bluetooth® LE. These unique features make PSoC™ 4 the industry’s most flexible and scalable low-power mixed-signal architecture. The PSoC™ 4 devices are classified as different families, as shown in Table 2, based on different features.

Table 2 PSoC™ 4 families

Classification	Family	Features	Details
Entry level	PSoC™ 4000 family	CAPSENSE™	Table 3
Intelligent analog	PSoC™ 4100 family	CAPSENSE™ + Programmable Analog	Table 4

PSoC™ 4 feature set

Classification	Family	Features	Details
Programmable digital	PSoC™ 4200 family	CAPSENSE™ + Programmable Analog + Programmable Digital Blocks	Table 5
Application specific	PSoC™ 4500 family	CAPSENSE™ + Motor Control	Table 6
	PSoC™ 4700 family	CAPSENSE™ + Inductive Sensing	
Analog coprocessor ²	PSoC™ 4A00 family	CAPSENSE™ + Programmable Analog Blocks	AN211293

Note: In [Table 3](#) and [Table 4](#), the columns highlighted in green indicate the family is supported in ModusToolbox™.

Table 3 PSoC™ 4000 family features

Features	PSoC™ 4000	PSoC™ 4000S	
CPU	16-MHz Cortex®-M0	48-MHz Cortex®-M0+	
Flash memory	16 KB	32 KB	
SRAM	2 KB	4 KB	
GPIOs	20	36	
CAPSENSE™	16 sensors	35 sensors	
Single-slope ADC (10-bit 46-ksps)	None	1	
Comparators	1 CSD comparator with a fixed threshold (1.2 V)	Two low-power comparators with wakeup feature	
IDACs ³	One 7-bit and one 8-bit	Two 7-bits	
Smart I/O ports	None	2	
Power supply range	1.71 V to 5.5 V	1.71 V to 5.5 V	
Low-power modes	Deep Sleep at 2.5 µA	Deep Sleep at 2.5 µA	
Segment LCD drive	None	4 COM segment LCD drive	
Serial communication	One I ² C	Two SCBs with programmable I ² C, SPI, or UART	
Timer Counter Pulse-Width Modulator (TCPWM)	1	5	
Clocks	Internal main oscillator (IMO)	24 MHz/32 MHz	24 MHz to 48 MHz
	Internal low-speed oscillator (ILO)	32-kHz internal ILO	40 kHz
	Watch crystal oscillator (WCO)	None	32-kHz
Power supply monitoring	Power-on reset (POR) Brown-out detection (BOD)	POR, BOD	
Supported kit	CY8CKIT-040 pioneer kit	CY8CKIT-041 pioneer kit	
Supported IDE	PSoC™ Creator	PSoC™ Creator, ModusToolbox™	

² See [AN211293](#) for getting started with PSoC™ 4 analog coprocessor family devices.

³ IDACs are available only when CAPSENSE™ is not in use. See the respective PSoC™ 4 architecture TRM for more details.

Table 4 PSoC™ 4100 family features

Features	PSoC™ 4100	PSoC™ 4100S	PSoC™ 4100S Plus	PSoC™ 4100S Plus 256K	PSoC™ 4100PS	PSoC™ 4100M	PSoC™ 4100 BL ⁴	PSoC™ 4100S Max
CPU	24-MHz Cortex®-M0	48-MHz Cortex®-M0+	48-MHz Cortex®-M0+	48-MHz Cortex®-M0+	48-MHz Cortex®-M0+	24-MHz Cortex®-M0	24-MHz Cortex®-M0	48-MHz Cortex®-M0+
DMA	N/A	N/A	8 channels	8 channels	8 channels	8 channels	8 channels	16 channels
Flash memory	32 KB	64 KB	128 KB	256 KB	32 KB	128 KB	256 KB	384 KB
SRAM	4 KB	8 KB	16 KB	32 KB	4 KB	16 KB	32 kB	32 KB
GPIOs	36	36	54	54	38	55	36	84
CAPSENSE™	1 channel, 35 sensors	1 channel, 35 sensors	1 channel, 53 sensors	1 channel, 53 sensors	1 channel, 33 sensors	2 channels, 54 sensors	1 channel, 35 sensors	2 channels, 80 sensors (32 control mux)
12-bit SAR ADC with sequencer	806-KSPS	1-MSPS	1-MSPS	1-MSPS	1-MSPS	806-KSPS	806-KSPS	1-MSPS
Opamps (programmable)	2	2	2	2	4/PGA	4	2	2
Programmable Voltage Reference (PVref)	None	None	None	None	Four channels	None	None	None
Voltage DAC (VDAC)	None	None	None	None	Two 13-bit VDAC	None	None	None
Comparators (low power with wakeup feature)	2	2	2	2	2	2	2	2
IDACs ⁵	One 7-bit and one 8-bit	Two 7-bits	Two 7-bits	Two 7-bits	Two 7-bits	Two 7-bits and two 8-bits	One 7-bit and one 8-bit	None
Smart I/O ports	None	2	3	2	1	None	None	3

⁴ See [AN91267](#) for getting started with PSoC™ 4 Bluetooth® LE Family devices.

⁵ IDACs are available only when CAPSENSE™ is not in use. See the respective PSoC™ 4 architecture TRM more details.

Features		PSoC™ 4100	PSoC™ 4100S	PSoC™ 4100S Plus	PSoC™ 4100S Plus 256K	PSoC™ 4100PS	PSoC™ 4100M	PSoC™ 4100 BL ⁴	PSoC™ 4100S Max
Power supply range		1.71 V to 5.5 V	1.71 V to 5.5 V	1.71 V to 5.5 V	1.71 V to 5.5 V	1.71 V to 5.5 V	1.71 V to 5.5 V	1.71 V to 5.5 V	1.71 V to 5.5 V
Low-power modes	Deep sleep	1.3 µA	2.5 µA	2.5 µA	2.5 µA	2.5 µA	1.35 µA	2.5 µA	2.5 µA
	Hibernate	150 nA	NA	NA	NA	NA	150 nA	NA	NA
	Stop	20 nA	NA	NA	NA	NA	35 nA	NA	NA
Segment LCD drive		4 COM	4 COM	4 COM	4 COM	4 COM	4 COM	4 COM	4 COM
SCBs with programmable I ² C, SPI, or UART		2	3	5	5	3	4	2	5
TCPWM		4	5	8	8	8	8	4	8
CAN		None	None	1	None	None	None	None	1
BLE		None	None	None	None	None	None	4.1/4.2	None
Clocks	IMO	3 MHz to 24 MHz	24 MHz to 48 MHz	24 MHz to 48 MHz	24 MHz to 48 MHz	24 MHz to 48 MHz	3 MHz to 48 MHz	24 MHz to 48 MHz	24 MHz to 48 MHz
	ILO	32 kHz	40 kHz	40 kHz	40 kHz	40 kHz	32 kHz	40 kHz	40 kHz
	WCO	Nil	32 kHz	32 kHz	32 kHz	32 kHz	32 kHz	32 kHz	32 kHz
Power supply monitoring		POR, BOD, Low-voltage detection (LVD)	POR, BOD	POR, BOD	POR, BOD	POR, BOD	POR, BOD, LVD	POR, BOD, LVD	POR, BOD
Supported kit		CY8CKIT-049 prototyping kit	CY8CKIT-041 pioneer kit	CY8CKIT-149 prototyping kit	--	CY8CKIT-147 prototyping kit	CY8CKIT-044 pioneer kit	CY8CKIT-042 BLE pioneer kit	CY8CKIT-041S-MAX pioneer kit

Features	PSoC™ 4100	PSoC™ 4100S	PSoC™ 4100S Plus	PSoC™ 4100S Plus 256K	PSoC™ 4100PS	PSoC™ 4100M	PSoC™ 4100 BL ⁴	PSoC™ 4100S Max
Supported IDE	PSoC™ Creator	PSoC™ Creator, ModusToolbox™	PSoC™ Creator, ModusToolbox™	PSoC™ Creator, ModusToolbox™	PSoC™ Creator	PSoC™ Creator	PSoC™ Creator	ModusToolbox™

PSoC™ 4 feature set

Table 5 PSoC™ 4200 family features

Features		PSoC™ 4200	PSoC™ 4200DS	PSoC™ 4200M	PSoC™ 4200L	PSoC™ 4200 BL ⁶
CM0 CPU		48 MHz Cortex®-M0	48 MHz Cortex®-M0	48 MHz Cortex®-M0	48 MHz Cortex®-M0	48 MHz Cortex®-M0
DMA		None	8 channels	8 channels	32 channels	None
Flash memory		32 KB	64 KB	128 KB	256 KB	256 KB
SRAM		4 KB	8 KB	16 KB	32 KB	32 KB
GPIOs		36	21	55	96	36
CAPSENSE™		1 channel, 35 sensors	None	2 channels, 54 sensors	2 channels, 94 sensors	1 channel, 35 sensors
ADC (12-bit, 1-MSPS SAR ADC with sequencer)		1	None	1	1	1
Opamps (programmable)		2	None	2	4	2
Comparators (low power with wakeup feature)		2	2	2	2	2
IDACs ⁷		One 7-bits and one 8-bit	None	Two 7-bits and two 8-bits	Two 7-bits and two 8-bits	One 7-bit and one 8-bit
Programmable logic blocks (UDBs)		4	4	4	8	4
Smart I/O ports		None	1	None	None	None
Power supply range		1.71 V to 5.5 V	1.71 V to 5.5 V	1.71 V to 5.5 V	1.71 V to 5.5 V	1.71 V to 5.5 V
Low-power modes	Deep Sleep	1.3 µA	2 µA	1.3 µA	1.3 µA	1.5 µA
	Hibernate	150 nA	NA	150 nA	150 nA	150 nA
	Stop	20 nA	NA	20 nA	20 nA	20 nA
Segment LCD drive		4 COM	None	4 COM	8 COM	4 COM
SCBs with programmable I ² C, SPI, or UART		2	3	4	4	2
TCPWM		4	4	8	8	4
CAN		None	None	2	2	None
BLE		None	None	None	None	4.1/4.2
USB Full Speed Device Controller (USB)		None	None	None	Yes	None
Clocks	IMO	3 MHz to 48 MHz	3 MHz to 48 MHz	3 MHz to 48 MHz	3 MHz to 48 MHz	3 MHz to 48 MHz
	ILO	32 kHz	40 kHz	32 kHz	32 kHz	32 kHz
	WCO	None	None	32 kHz	32 kHz	32 kHz

⁶ See [AN91267](#) for getting started with PSoC™ 4 Bluetooth® LE family devices.

⁷ IDACs are available only when CAPSENSE™ is not in use. See the respective PSoC™ 4 architecture TRM for more details.

PSoC™ 4 feature set

Features		PSoC™ 4200	PSoC™ 4200DS	PSoC™ 4200M	PSoC™ 4200L	PSoC™ 4200 BL ⁶
	External crystal oscillator (ECO)	None	None	4 MHz to 33 MHz	None	None
Power supply monitoring		POR, BOD, LVD	POR, BOD	POR, BOD, LVD	POR, BOD, LVD	POR, BOD, LVD
Supported kit		CY8CKIT-042 pioneer kit	CY8CKIT-146 prototyping kit	CY8CKIT-044 pioneer kit	CY8CKIT-046 pioneer kit	CY8CKIT-042 Bluetooth® LE pioneer kit
Supported IDE		PSoC™ Creator	PSoC™ Creator	PSoC™ Creator	PSoC™ Creator	PSoC™ Creator

Table 6 PSoC™ 4500 and PSoC™ 4700 family features

Features		PSoC™ 4500S	PSoC™ 4700S
CM0+ CPU		48 MHz Cortex®-M0+	48 MHz Cortex®-M0+
DMA		8 channels	None
Flash memory		256 KB	32 KB
SRAM		32 KB	4 KB
GPIOs		53	36
CAPSENSE™		1 channel, 52 sensors	1 channel, 35 sensors
MagSense		None	1 channel
ADC		Two 12-bits, 1-MSPS SAR ADCs with sequencer	10-bit, 16.8-ksps Single slope ADC
Opamps (programmable)		4	None
Comparators (low power with wakeup feature)		2	2
IDACs ⁸		Two 7-bits	Two 7-bits
Smart I/O ports		2	2
Power supply range		1.71 V to 5.5 V	1.71 V to 5.5 V
Low-power modes	Deep Sleep	1.3 µA	2.5 µA
	Hibernate	150 nA	NA
	Stop	20 nA	NA
Segment LCD drive		4 COM	8 COM
SCBs with programmable I ² C, SPI, or UART		5	2
TCPWM		8	5
Motor Control Acceleration (MCA)		2	None

⁸ IDACs are available only when CAPSENSE™ is not in use. See the respective PSoC™ 4 architecture TRM for more details.

PSoC™ 4 feature set

Features		PSoC™ 4500S	PSoC™ 4700S
Clocks	IMO	24 MHz to 48 MHz	24 MHz to 48 MHz
	ILO	40 kHz	40 kHz
	WCO	32 kHz	32 kHz
	ECO	4 MHz to 33 MHz	None
Power supply monitoring		POR, BOD	POR, BOD
Supported kit		CY8CKIT-045S pioneer kit	CY8CKIT-148 evaluation kit
Supported IDE		PSoC™ Creator	PSoC™ Creator

PSoC™ is more than an MCU

4 PSoC™ is more than an MCU

Figure 10 shows that a typical MCU contains a CPU (such as 8051 or an Arm® Cortex®) with a set of peripheral functions such as ADCs, DACs, UARTs, SPIs, and general I/O, all linked to the CPU’s register interface. Within the MCU, the CPU is the “heart” of the device – the CPU manages everything from setup to data movement to timing. Without the CPU, the MCU cannot function.

Figure 11 shows that PSoC™ is quite different. With PSoC™, the CPU, analog, digital, and I/O are equally important resources in a programmable system. *It is the system’s interconnect and programmability that is the heart of PSoC™ – not the CPU.* The peripheral analog and digital are interconnected with a highly configurable matrix of signal and data bus meshing that allows you to create custom designs that meet your application requirements. *You can program PSoC™ to emulate an MCU, but you cannot program an MCU to emulate PSoC™.*

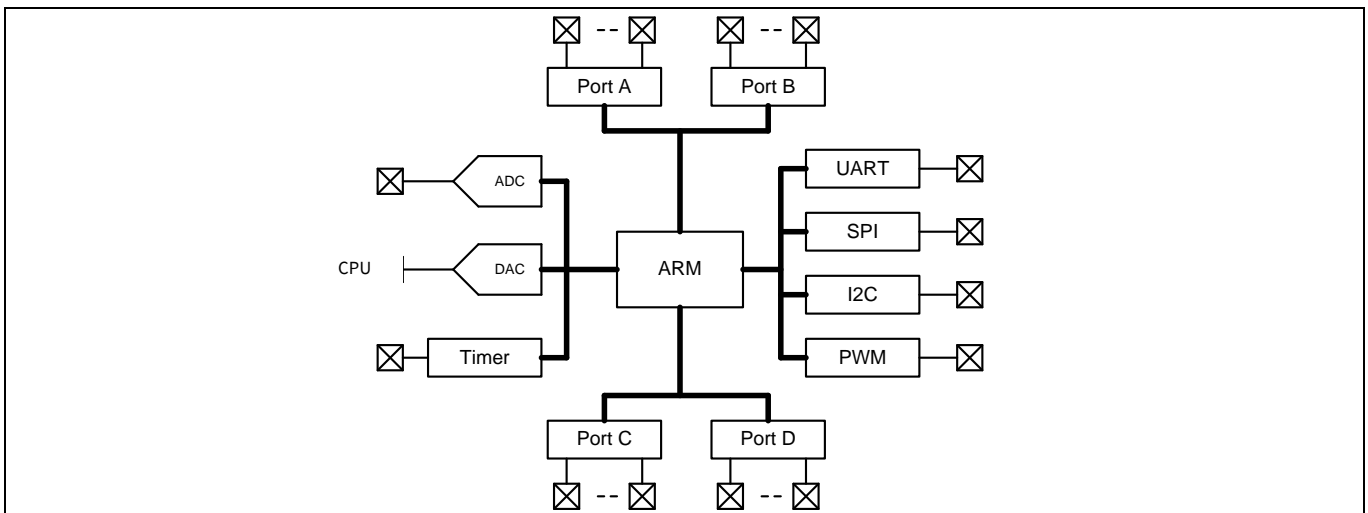


Figure 10 Block diagram of a typical MCU

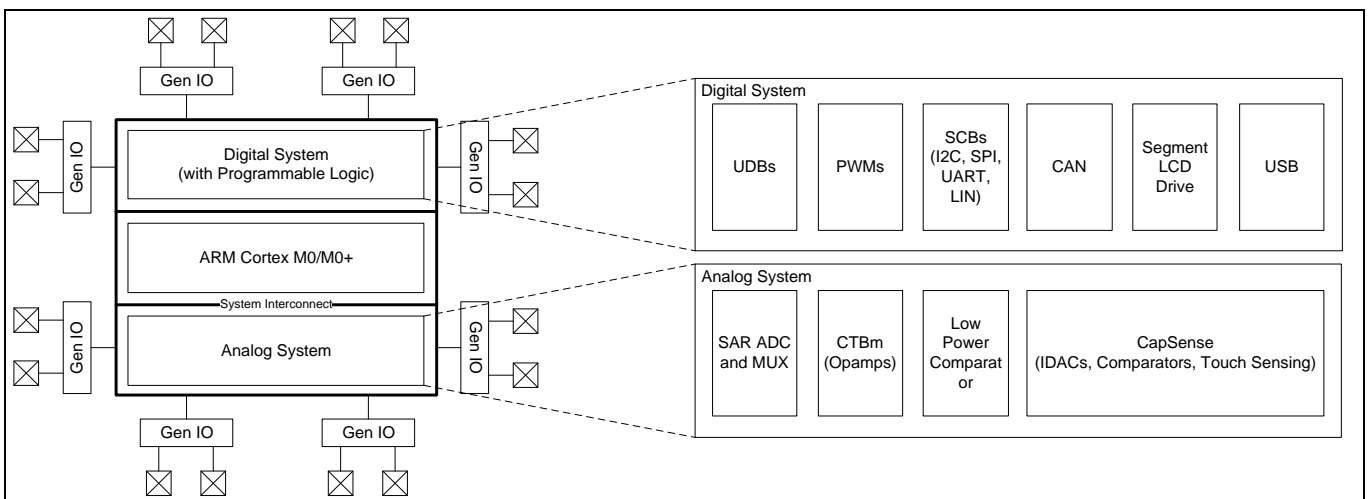


Figure 11 PSoC™ block diagram

A typical MCU requires CPU firmware to process state machines, use a timer for timing, and drive an output pin. Thus, the functional path is almost always through the CPU. However, with PSoC™, asynchronous parallel processing is possible. You can configure a PSoC™ to have elements that operate independently from the CPU. The projects included with this application note demonstrate this concept. The PSoC™ is configured to make an LED blink without writing any code for the CPU.

PSoC™ is more than an MCU

4.1 The concept of PSoC™ Creator Components

One other important thing about PSoC™ is the availability of PSoC™ Creator IDE. In PSoC™ Creator, different PSoC™ resources are organized as graphical elements called Components, which can be dragged and dropped onto a schematic to quickly build designs. Every peripheral in PSoC™ is available as a pre-validated PSoC™ Creator Component – PWM Component, ADC Component, DAC Component, CAPSENSE™ Component, UART Component and so on. The availability of pre-validated Components in the PSoC™ Creator significantly reduces the development time. It also allows you to quickly make changes in the design using graphical options.

For example, configuring a PWM to blink an LED in a typical microcontroller involves the following:

1. Locate the registers corresponding to the PWM block.
2. Calculate the values to be written to the PWM registers based on the required PWM period and duty cycle.
3. Write many lines of code to configure the PWM registers, set the pin drive mode, and to connect the PWM output to the pin. Many MCUs do not offer alternate pins to connect to the internal blocks.

To implement the same functionality in PSoC™ is a trivial exercise, as you will find out later in this application note. Later, if you need to reconfigure the same PWM block to a Timer, you do not need anything more than a few mouse clicks in PSoC™ Creator.

The PSoC™ also has programmable digital blocks known as Universal Digital Blocks (UDBs). PSoC™ Creator also provides several Components made of UDBs such as UART, SPI, I2S, Timer, PWM, Counter, Digital Gates (AND, OR, NOT, XOR, and so on), and many more. You can even create your own custom state machines and digital logic using the UDBs in PSoC™ Creator. The method to create your own custom PSoC™ Creator Components is provided in the [PSoC™ Creator Component author guide](#).

My first PSoC™ 4 design using ModusToolbox™

5 My first PSoC™ 4 design using ModusToolbox™

This section:

- Demonstrates how PSoC™ can be programmed to do more than a traditional MCU.
- Shows how to build a simple PSoC™ application and program it into a development kit.
- Provides detailed steps to use Eclipse IDE for **ModusToolbox™**.

However, ModusToolbox™ software is IDE-neutral. The steps in this section use the Eclipse IDE for ModusToolbox™ to launch the Project Creator. The resulting project is automatically imported into the IDE.

Note that you can use Project Creator, Library Manager, and any Configurator as stand-alone tools on Linux, macOS, or Windows. These tools create or modify files in your application folder. From the command line, you can then export that application to the supported third-party IDEs, like VS Code, IAR Embedded Workbench, or Keil µVision. It is not mandatory to use Eclipse IDE. For more details, see Exporting to IDEs in the **ModusToolbox™ user guide**.

5.1 Before you begin

5.1.1 Have you installed ModusToolbox™?

Download and install ModusToolbox™ from the **ModusToolbox™ home page**. After installing the software, see the Quick Start Guide and User Guide in ModusToolbox™ IDE to get an overview of the software.

5.1.2 Do you have a development kit or prototyping kit?

Testing this design requires one of the kits listed in **Table 7**, which has an integrated programmer.

Table 7 List of PSoC™ 4 pioneer kits, prototyping kits, and supported devices

Kit name	Kit type	Supported device family	Part number
CY8CKIT-145	Prototyping kit	PSoC™ 4000S	CY8C4045AZI-S413
CY8CKIT-041-41XX	Pioneer kit	PSoC™ 4100S	CY8C4146AZI-S433
CY8CKIT-149	Prototyping kit	PSoC™ 4100S Plus	CY8C4147AZI-S475
CY8CKIT-041S-Max	Pioneer kit	PSoC™ 4100S Max	CY8C4149AZI-S598

5.2 Using these instructions

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

- **Part 1: Creating a new application**
- **Part 2: Viewing and modifying the design**
- **Part 3: Writing firmware**
- **Part 4: Building the application**
- **Part 5: Programming the device**
- **Part 6: Testing your design**

This design is developed for the kits listed in **Table 7**. You can test this example by selecting the appropriate kit while creating the application.

My first PSoC™ 4 design using ModusToolbox™

5.3 About the design

This design uses the CM0+ CPU of PSoC™ 4 to execute two tasks: UART communication and LED control. The CM0+ CPU uses the UART to print “Hello World” message to the serial port stream and starts blinking the user LED on the kit.

5.4 Part 1: Creating a new application

This section provides the step-by-step process to create a new application. The ‘**Empty PSoC4 App**’ starter application is used to guide you through the design development stages and programming.

If you are familiar with developing projects with ModusToolbox™, you can use the ‘**Hello World**’ starter application directly; see [Code example](#) for details. This starter application is a complete design, with the firmware written for the supported kits. You can go through the instructions and observe how the steps are implemented in the code example.

Even if you start from the scratch and follow the instructions in this application note, you can use the code example as a reference.

Launch ModusToolbox™ to get started. Note that ModusToolbox™ will need access to the internet to successfully clone the starter application onto your machine.

5.4.1 Select a new workspace

When launched, ModusToolbox™ displays a dialog, where you can choose a directory for the workspace. The workspace directory is used to store workspace preferences and development artifacts. Click **Browse** and choose an existing empty directory, alternatively you can type a directory name along with the complete path, and ModusToolbox™ will create the directory.

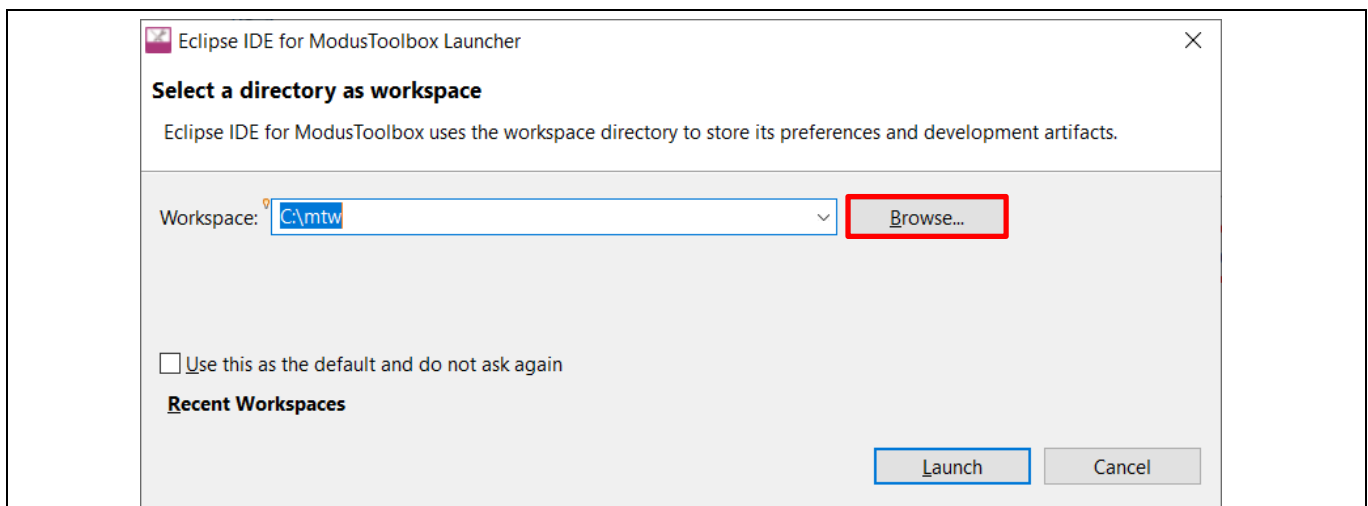


Figure 12 Selecting workspace directory

5.4.2 Create a new ModusToolbox™ application

- Click **New Application** in the **Start** group of the **Quick Panel**.
- Alternatively, choose **File > New > ModusToolbox Application**.

The ModusToolbox™ IDE application window appears.

My first PSoC™ 4 design using ModusToolbox™

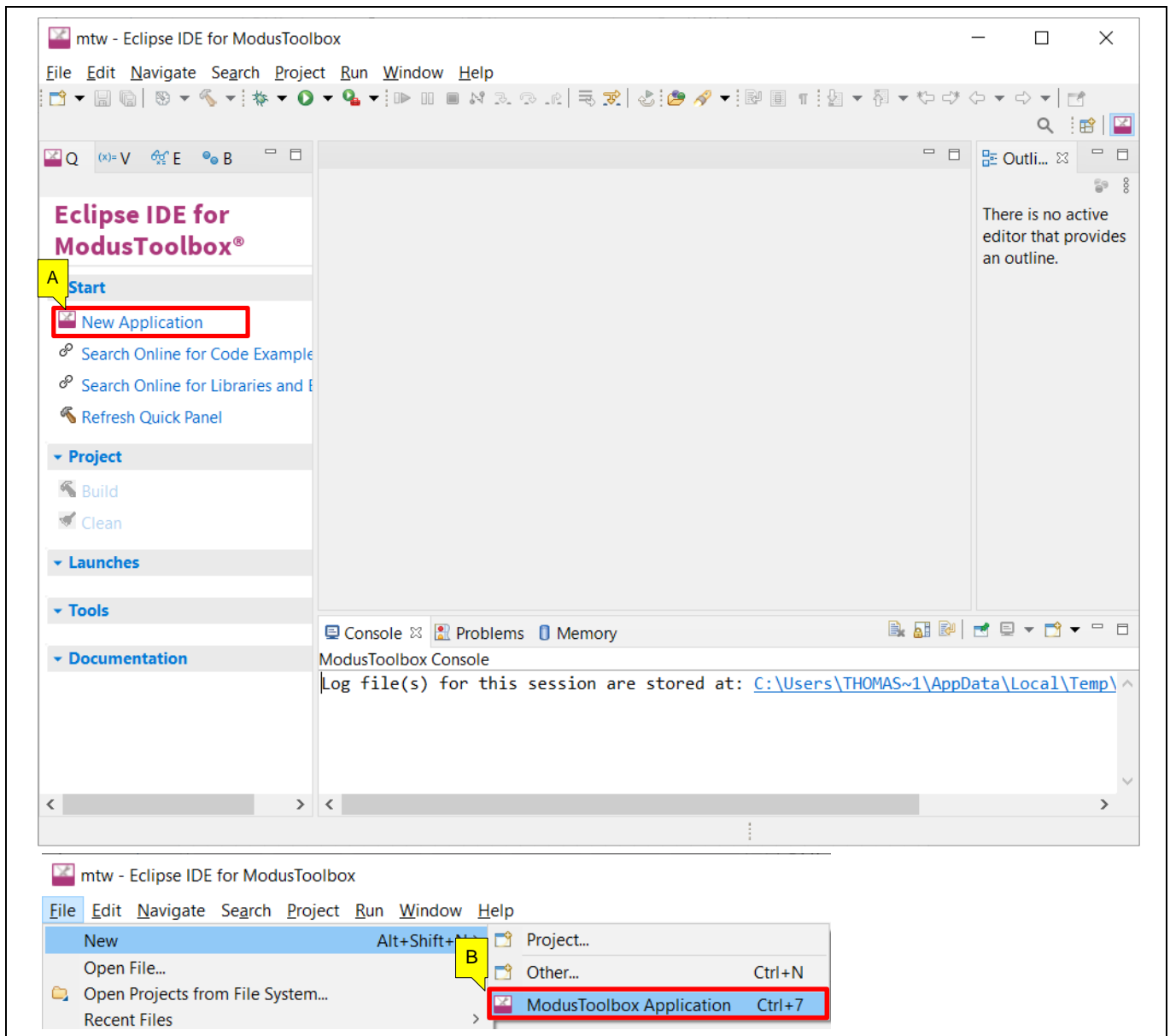


Figure 13 Create a new ModusToolbox™ IDE application

5.4.3 Select a target PSoC™ 4 development kit

ModusToolbox™ speeds up the development process by providing BSPs that set various workspace/project options for the development kit specified in the new application dialog.

- In the Choose Board Support Package (BSP) dialog, choose the Kit Name, for example, CY8CKIT-041S-MAX.
- Click **Next**.

My first PSoC™ 4 design using ModusToolbox™

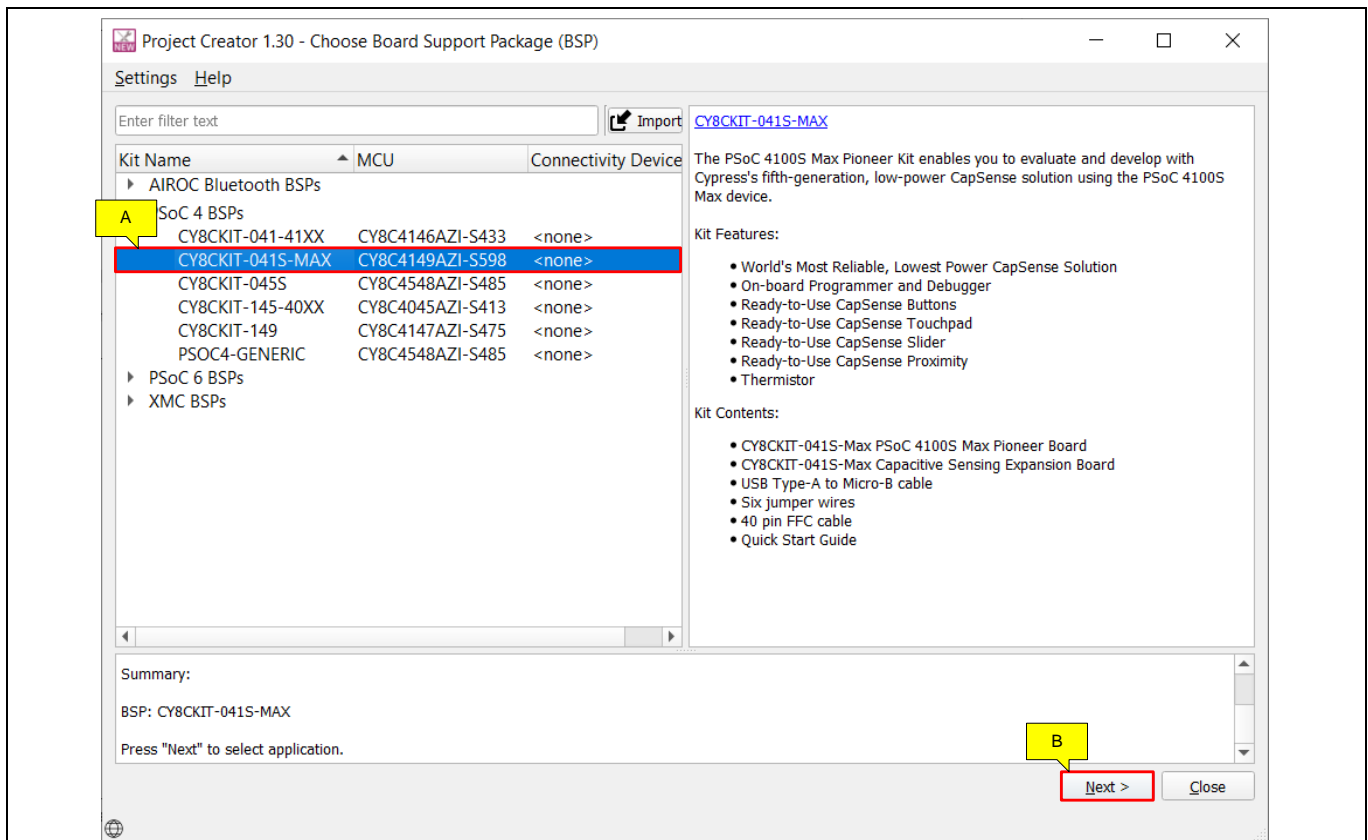


Figure 14 Choose target hardware

- In the **Select Application** dialog, select the **Empty PSoC4 App** template application.
- In the **New Application Name** field, type a name for the application, such as **Hello_World**. You can also retain the default name.
- Click **Create**. ModusToolbox™ will create the application project.

My first PSoC™ 4 design using ModusToolbox™

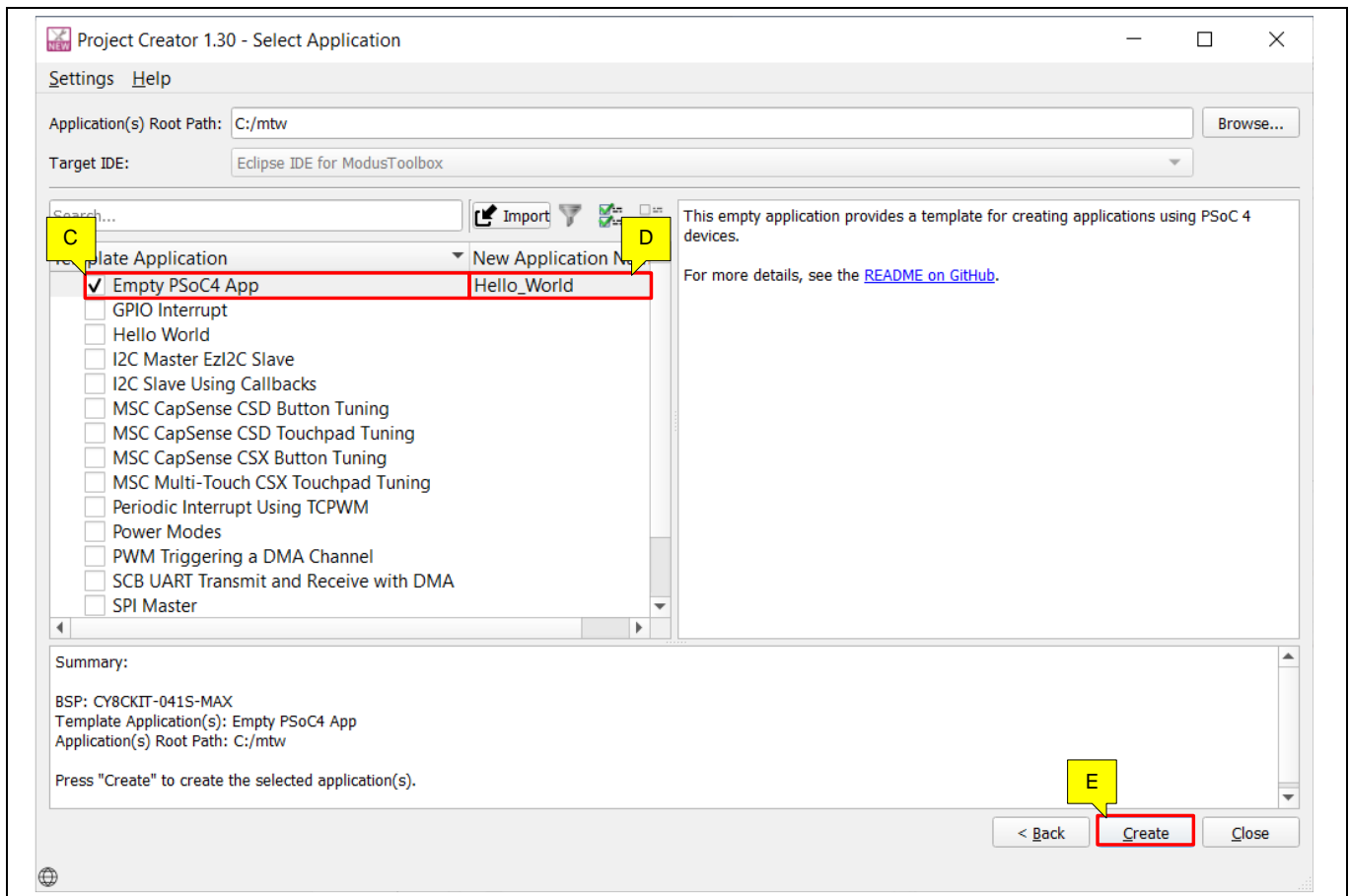


Figure 15 Choosing starter application

You have successfully created a new ModusToolbox™ application for a PSoC™ 4.

The BSP uses the device listed in [Table 7](#) based on the selected kit.

If you are using custom hardware based on PSoC™ 4, or a different PSoC™ 4 part number, see the [ModusToolbox™ user guide](#). The guide is also available in the `ide_2.2>docs` folder of the ModusToolbox™ installation directory.

5.5 Part 2: Viewing and modifying the design

5.5.1 Project structure

Figure 16 shows the ModusToolbox™ project explorer displaying the structure of the application project.

In the ModusToolbox™ IDE, a PSoC™ 4 application consists of a project to develop code for the CM0+ CPU.

- The project folder consists of various subfolders, each denoting a specific aspect of the project.
- The *build* folder contains all artifacts resulting from the make build of the project. The output files are organized by target BSPs.
- The *libs* folder has libraries that are local to the application. By default, the BSP for the created application is local and available in this folder. This allows you to change the BSP for a given application without worrying that the changes will propagate to other applications.

My first PSoC™ 4 design using ModusToolbox™

- d) The *mtb_shared* folder has sub-folders belonging to different middleware; PSoC™ 4 HAL, PSoC™ 4 PDL, CAPSENSE™ etc. These are individual libraries that are downloaded based on the *.mtb* files provided within the project. These libraries, by default, are shared across the projects in the workspace.
- e) The *.mtb* files provide the location from which ModusToolbox™ pulls the content. These files typically contain the GitHub location of the entire library. A *.mtb* file can point to content that contains another *.mtb* file. ModusToolbox™ processes this nested *.mtb* file recursively and downloads all libraries.

For example, the BSP lib file *TARGET_CY8CKIT-041S-MAX.mtb* points to https://github.com/Infineon/TARGET_CY8CKIT-041S-MAX#latest-v1.X. The *latest-v1.X* tag in the link denotes the specific release of the BSP.

In another example, *capsense.mtb* points to the library hosted at <https://github.com/Infineon/capsense#latest-v3.X>.

- f) Note that application project contains a *Makefile*. This has instructions on how to recreate the project. This file also contains the set of directives, which the make tool uses to compile and link the application project.

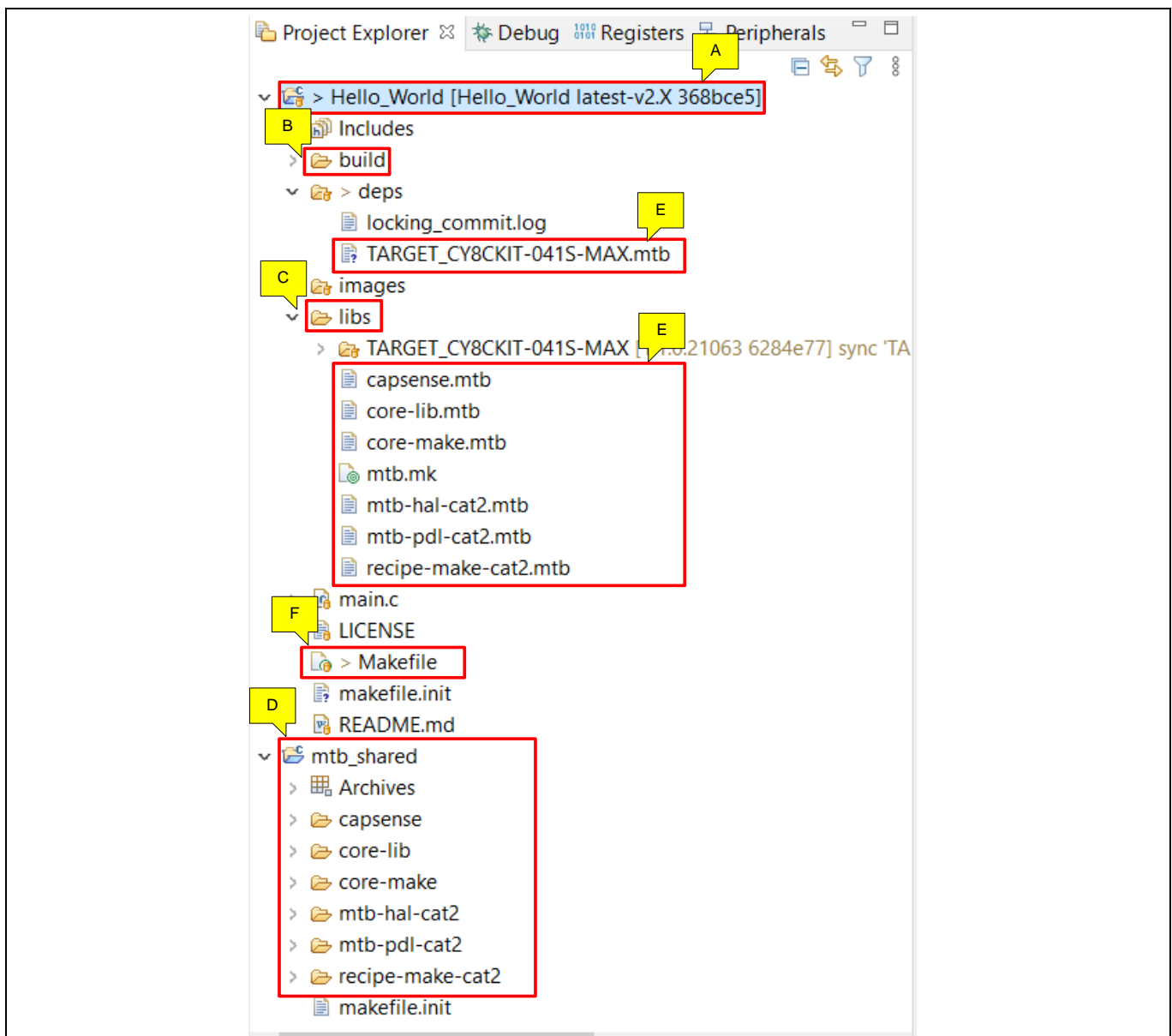


Figure 16 Project explorer view

My first PSoC™ 4 design using ModusToolbox™

- g) The files provided by the BSP are listed under the *TARGET_x* folder in the *libs* folder. All configuration files generated by the device and peripheral configurators are included in the *COMPONENT_BSP_DESIGN_MODUS/GeneratedSource* folder of the BSP and are prefixed with *cycfg_*. These files contain the design configuration defined by the BSP. Double-click the *design.modus* file to view the design configuration.

The *TARGET_x* folder also contains the *.lib* files in the *deps* folder that specify the middleware and other libraries used in the project, and the linker scripts and the startup code for the PSoC™ 4 device used on the board.

Figure 17 shows the ModusToolbox™ project explorer displaying the structure of the *libs* folder, which contains the BSP files for this workspace.

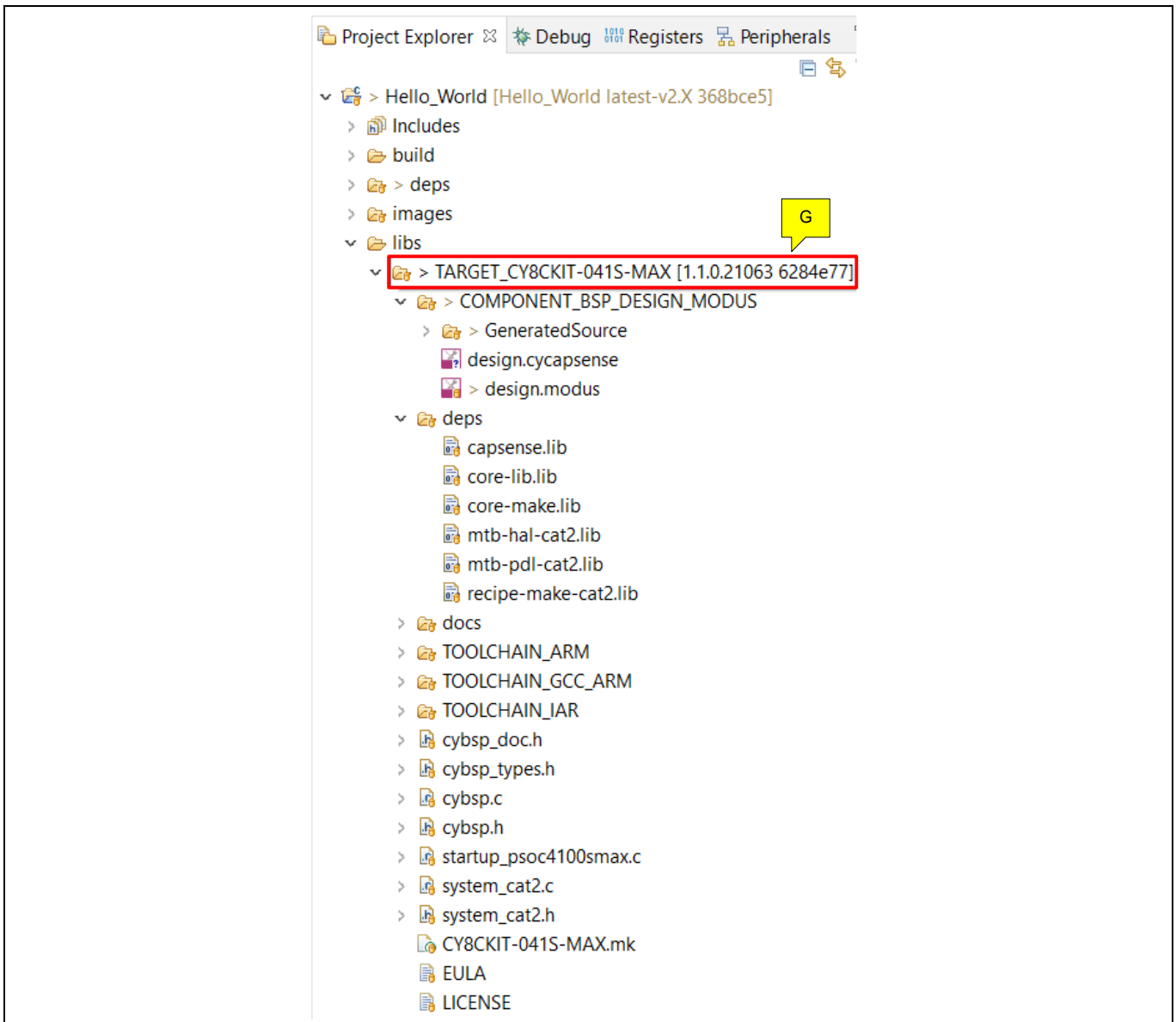


Figure 17 Project explorer view – *libs/TARGET_x* folder expanded

My first PSoC™ 4 design using ModusToolbox™

5.5.2 Modify the design

1. Double-click the *design.modus* file from the *libs>TARGET_X>COMPONENT_BSP_x* folder .

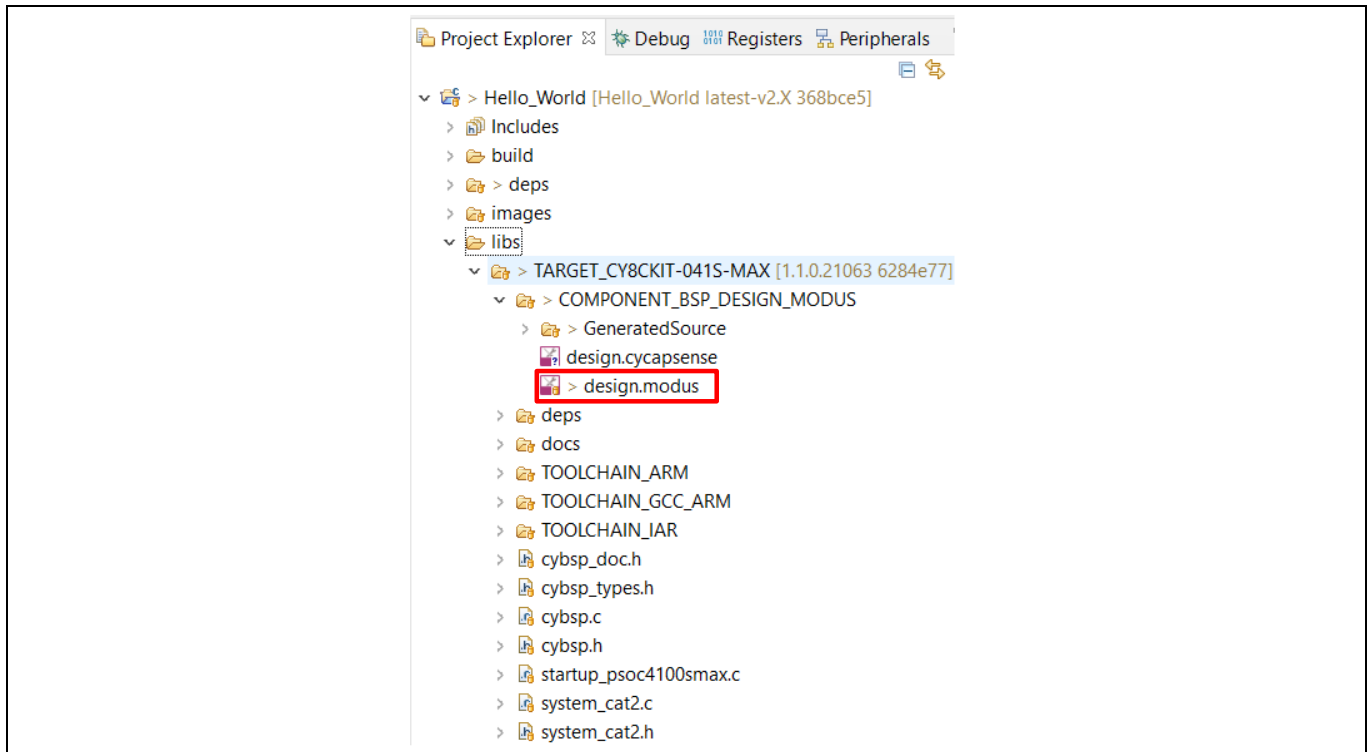


Figure 18 Opening the *design.modus* file

This opens the **Device Configurator** dialog. You can also double-click other *design.x* files to open them in their respective configurators or click the corresponding links in the **Quick Panel** and configure as required.

My first PSoC™ 4 design using ModusToolbox™

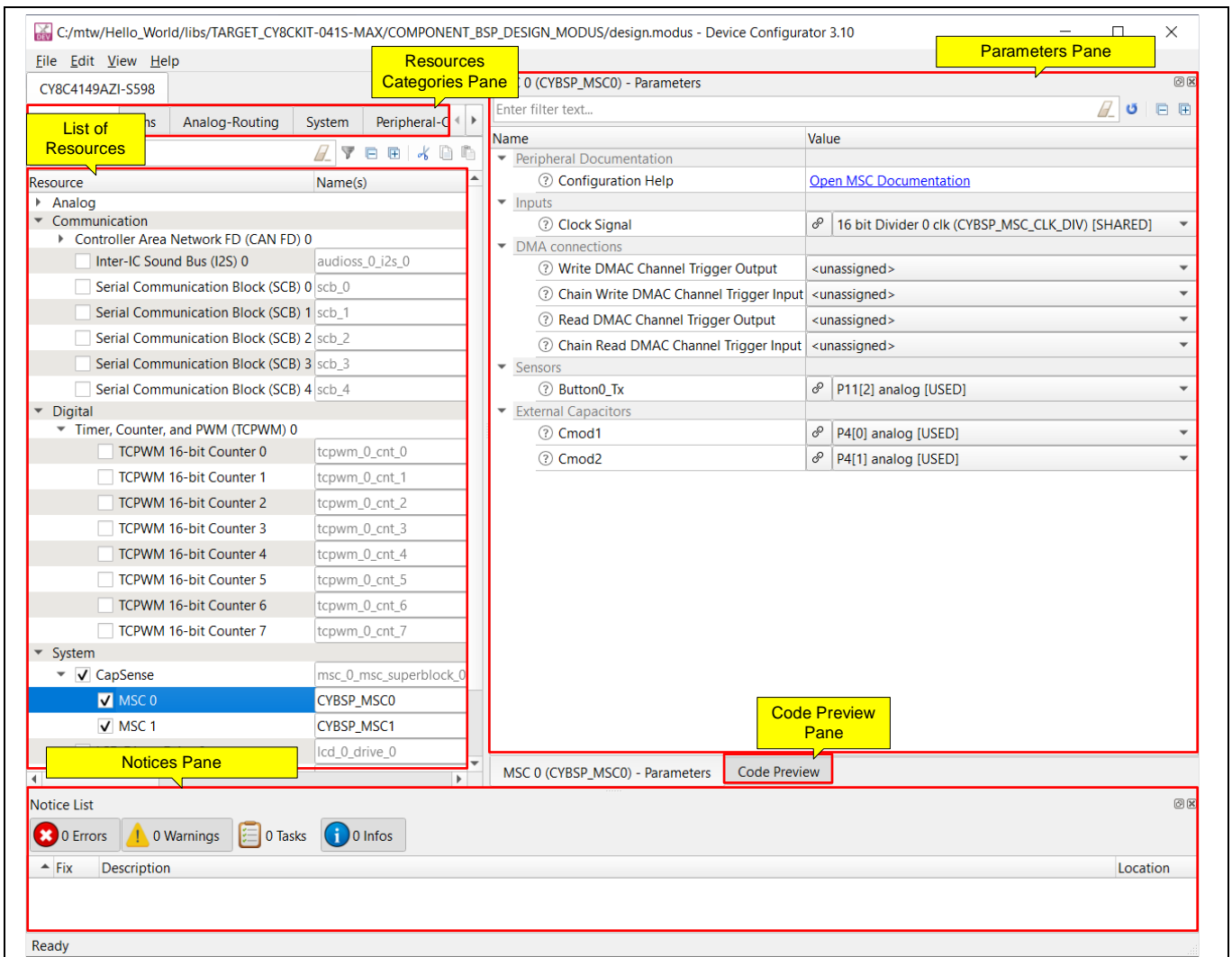


Figure 19 Overview of design.modus

From the **Resources Categories** pane, of the **Device Configurator** dialog, you can choose from the different resources, such as peripherals, pins, and clocks, available for the device. The **Peripherals** tab shows a **List of Resources** available in the device.

The **Personality** defines the behavior of the resource. For example, a **Serial Communication Block (SCB)** resource can have an **EZ12C**, **I2C**, **SPI**, or **UART** personalities. The **Name(s)** field is the name of the resource, which is used in firmware development. You can specify one or more names separated by a comma (with no space).

In the **Parameters** pane you can enter the configuration parameters for each enabled resource and the selected personality. The **Code Preview** pane shows the configuration code generated for the selected configuration parameters. This code is populated in the `cycfg_` files in the `GeneratedSource` folder. Any errors, warnings, and information messages, caused by the configuration, are displayed in the **Notices** pane.

The application project contains relevant files that will help in creating an application for the CM0+ CPU (`main.c`). This C file is compiled and linked with the CM0+ image as part of the normal build process.

- Go to the **Peripheral-Clocks** tab in Resource Categories pane to configure the clock for UART component.

My first PSoC™ 4 design using ModusToolbox™

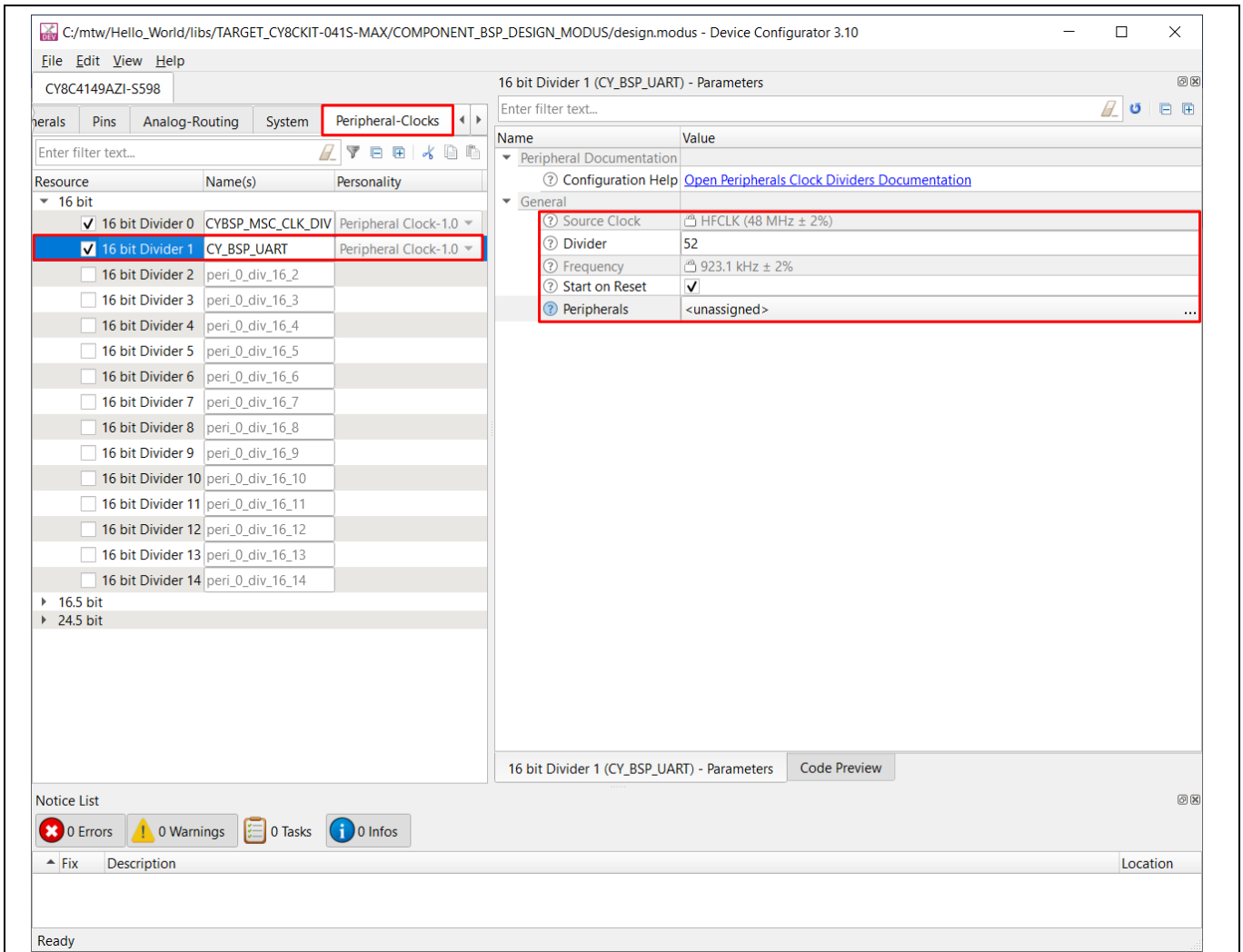


Figure 20 Configuring clock

- Go to the Peripherals tab in Resource Categories pane and enable Serial Communication Block (SCB) 1, set personality as UART-1.0 and the name as CYBSP_UART. In Figure 21 SCB instance 1 is selected based on Rx/Tx pin required for the kit. See the [device datasheet](#) for the pins available for each instance.

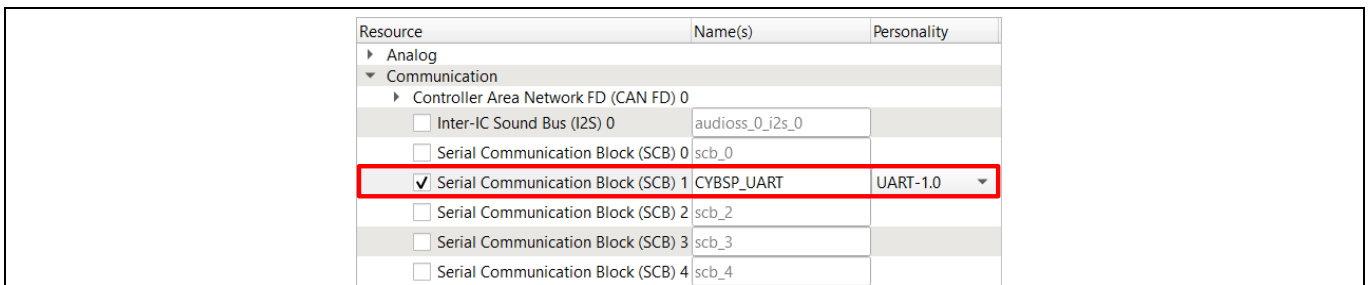


Figure 21 Enabling SCB as UART

- Set the required configuration parameters such as **Clock**, **Rx**, and **Tx** in the **Parameter** pane. Retain the default values for other parameters.

My first PSoC™ 4 design using ModusToolbox™

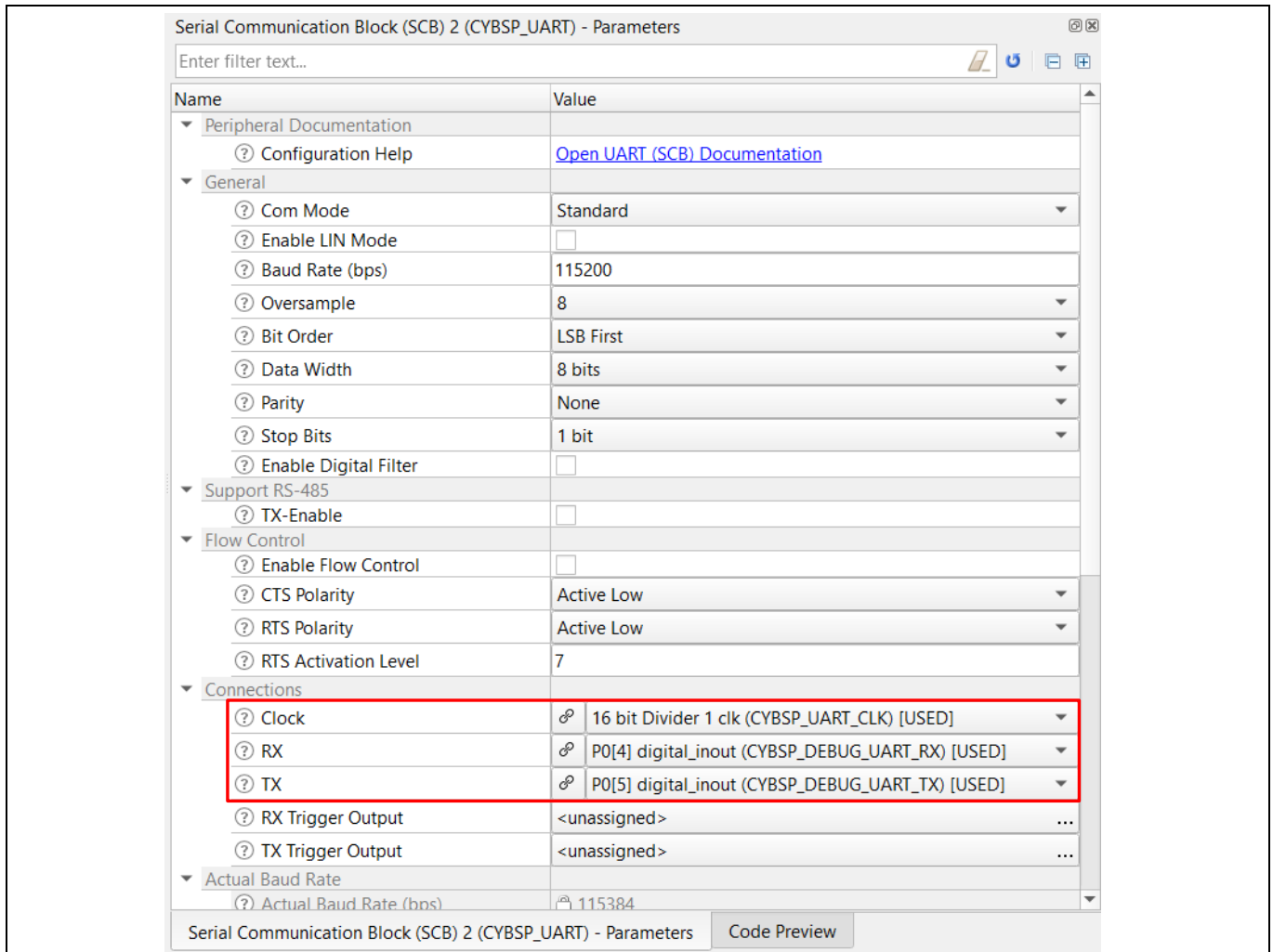
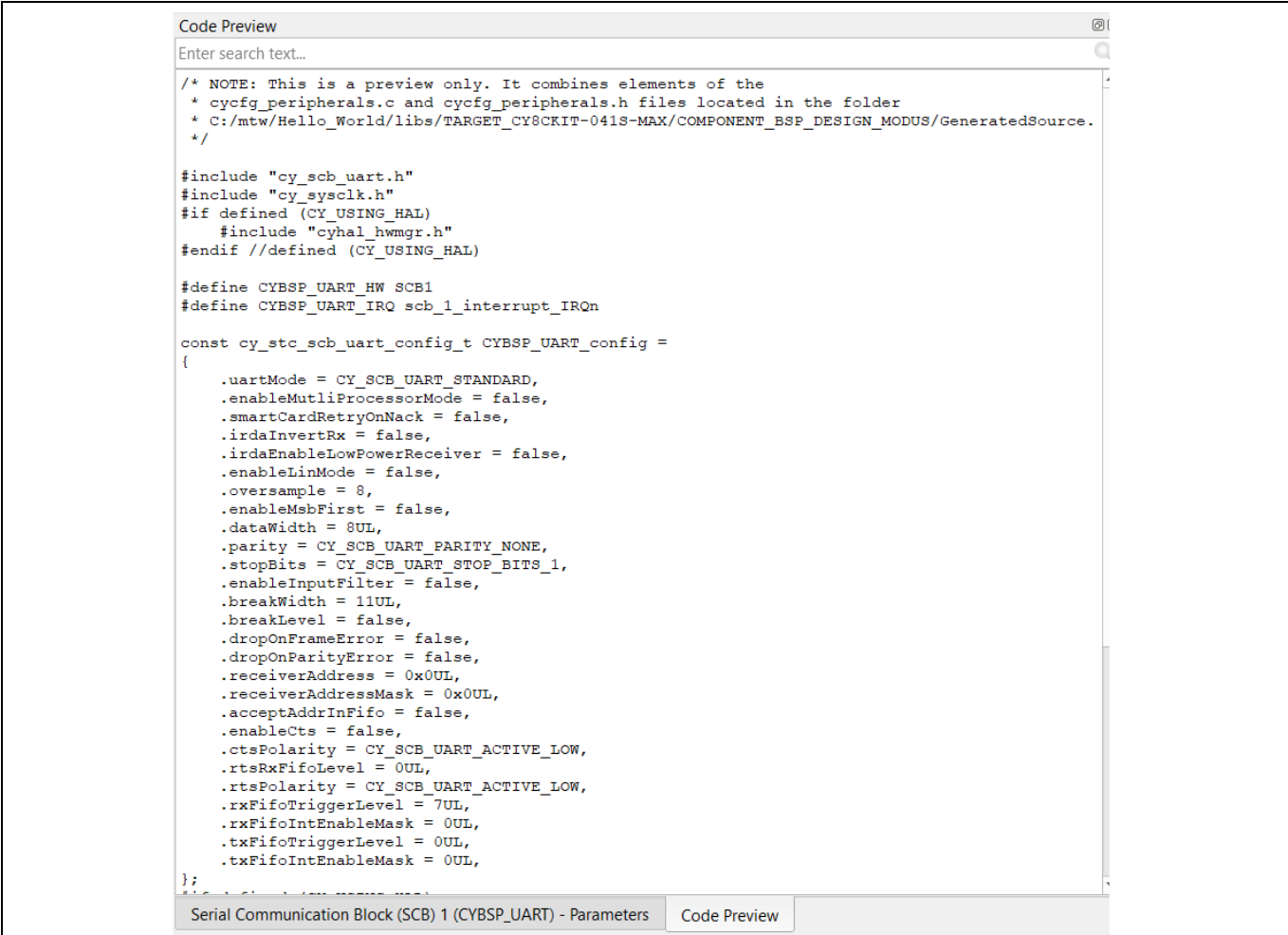


Figure 22 Setting UART configuration parameter

5. Go to the **Code Preview** pane to preview the generated code.

My first PSoC™ 4 design using ModusToolbox™



```

Code Preview
Enter search text..

/* NOTE: This is a preview only. It combines elements of the
 * cycfg_peripherals.c and cycfg_peripherals.h files located in the folder
 * C:/mtw/Hello_World/libs/TARGET_CY8CKIT-041S-MAX/COMPONENT_BSP_DESIGN_MODUS/GeneratedSource.
 */

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

#define CYBSP_UART_HW SCB1
#define CYBSP_UART_IRQ scb_1_interrupt_IRQn

const cy_stc_scb_uart_config_t CYBSP_UART_config =
{
    .uartMode = CY_SCB_UART_STANDARD,
    .enableMutliProcessorMode = false,
    .smartCardRetryOnNack = false,
    .irdaInvertRx = false,
    .irdaEnableLowPowerReceiver = false,
    .enableLinMode = false,
    .oversample = 8,
    .enableMsbFirst = false,
    .dataWidth = 8UL,
    .parity = CY_SCB_UART_PARITY_NONE,
    .stopBits = CY_SCB_UART_STOP_BITS_1,
    .enableInputFilter = false,
    .breakWidth = 11UL,
    .breakLevel = false,
    .dropOnFrameError = false,
    .dropOnParityError = false,
    .receiverAddress = 0x0UL,
    .receiverAddressMask = 0x0UL,
    .acceptAddrInFifo = false,
    .enableCts = false,
    .ctsPolarity = CY_SCB_UART_ACTIVE_LOW,
    .rtsRxFifoLevel = 0UL,
    .rtsPolarity = CY_SCB_UART_ACTIVE_LOW,
    .rxFifoTriggerLevel = 7UL,
    .rxFifoIntEnableMask = 0UL,
    .txFifoTriggerLevel = 0UL,
    .txFifoIntEnableMask = 0UL,
};

```

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

Figure 23 Code preview pane

- Go to the **Pins** tab in **Resource Categories** pane to configure the GPIO to make the user LED on the kit blink. See [Table 8](#) if you are using a different PSoC™ 4 kit.

My first PSoC™ 4 design using ModusToolbox™

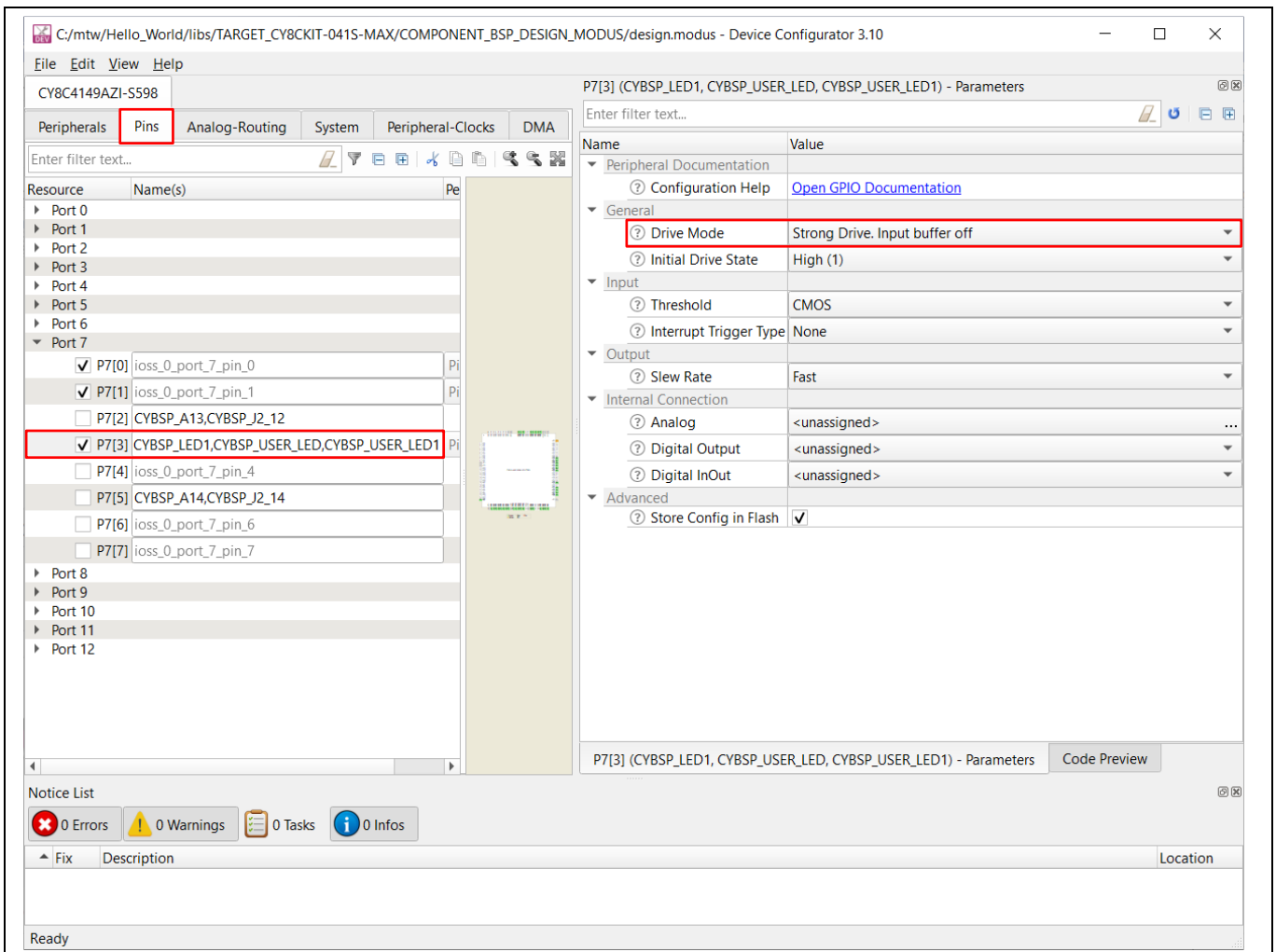


Figure 24 Configuring GPIO

Table 8 Pin mapping table across PSoC™ 4 kits

Function	CY8CKIT-145 (PSoC™ 4000S)	CY8CKIT-041-41XX (PSoC™ 4100S)	CY8CKIT-149 (PSoC™ 4100S Plus)	CY8CKIT-041S-MAX (PSoC™ 4100S Max)
User LED	P2[5]	P3[4]	P3[4]	P7[3]

- Go to the different tabs in the **Resource Categories** pane to view and configure other resources. For this design no further change is required.
- Follow the path **File** → **Save** or press **[Ctrl]+[S]** to save the configuration and to generate the source code in the *GeneratedSource* folder.

Note: In most cases, before doing any real design work on your application, you should create a BSP for your device or board. This allows you to configure the settings for your own custom hardware or for different linker options. Also, you can save the BSP to be used in future applications. See *Creating a BSP for Your Board* in the [ModusToolbox™ user guide](#) for more details.

5.6 Part 3: Writing firmware

This design uses the CM0+ CPU of the PSoC™ 4 to execute two tasks: UART communication and LED control. The CM0+ CPU uses the UART to print the “Hello World” message to the serial port stream and to make the user LED on the kit blink.

My first PSoC™ 4 design using ModusToolbox™

If you are using the **Empty PSoC™ 4** starter application, you can copy the respective source code, from the code snippet provided in this section, to the *main.c* file of the application project. If you are using the **Hello World** code example, the required files are already in the application.

5.6.1 Firmware flow

This section explains the code in the *main.c* file of the application.

In this example, the CM0+ CPU comes out of reset and performs resource initialization. It configures the system clocks, pins, clock to peripheral connections, and other platform resources.

The clocks and system resources are initialized by the BSP initialization function. PDL functions are used to configure and enable the UART Component. The UART prints “Hello World” message on the terminal emulator – the onboard KitProg3 acts as the USB-UART bridge to create the virtual COM port. An infinite FOR loop, with a software delay, is used to toggle the user LED periodically.

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

- `cybsp_init()` - This BSP function initializes the system resources of the device including but not limited to the system clocks and power regulators.
- `Cy_SCB_UART_Init()` - This PDL function initializes the SCB block for UART operation. A configuration structure, `CYBSP_UART_config`, is used as a parameter for this function to configure the UART. This structure is auto-generated by *design.modus* based on the applied configuration.
- `Cy_SCB_UART_Enable()` - This PDL function enables the SCB block for UART operation.
- `Cy_SCB_UART_PutString()` - This PDL function places a NULL terminated string in the UART TX FIFO.
- `Cy_GPIO_Inv()` - This PDL function sets a pin output logic state to the inverse of the current output logic state.
- `Cy_SysLib_Delay()` - This PDL function delays by the specified number of milliseconds.

Copy **Code Listing 1** to *main.c* of your application project.

Code Listing 1 Code snippet *main.c*

```
#include "cy_pdl.h"
#include "cybsp.h"

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

/*****
 * Function Name: main
 *****/
 * Summary:
 * System entrance point. This function performs
 * - initial setup of device
```

My first PSoC™ 4 design using ModusToolbox™

```

* - configure the SCB block as UART interface
* - prints out "Hello World" via UART interface
* - Blinks an LED under firmware control at 1 Hz
*
* Parameters:
* none
*
* Return:
* int
*
*****
****/
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();

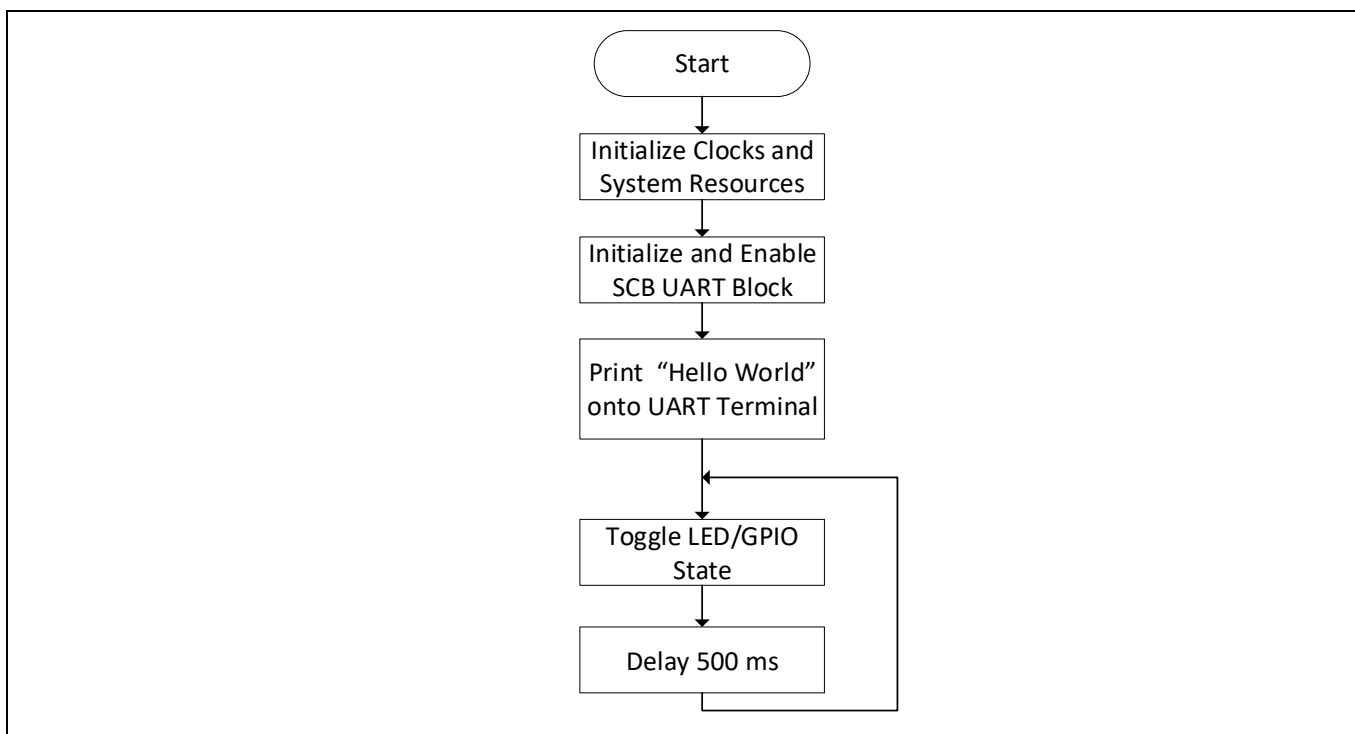
    /* Send a string over serial terminal */
    Cy_SCB_UART_PutString(CYBSP_UART_HW, "Hello world\r\n");

    for (;;)
    {
        /* Toggle the user LED state */
        Cy_GPIO_Inv(CYBSP_USER_LED1_PORT, CYBSP_USER_LED1_PIN);

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

```

My first PSoC™ 4 design using ModusToolbox™

**Figure 25** Firmware flowchart

This summarizes how the firmware works in the code example. Explore the source files for a deeper understanding.

5.7 Part 4: Building the application

This section shows how to build the application.

5.7.1 Build the application

- Select the application project in the Project Explorer window and click 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.
- The **Console** view lists the results of the build operation.

My first PSoC™ 4 design using ModusToolbox™

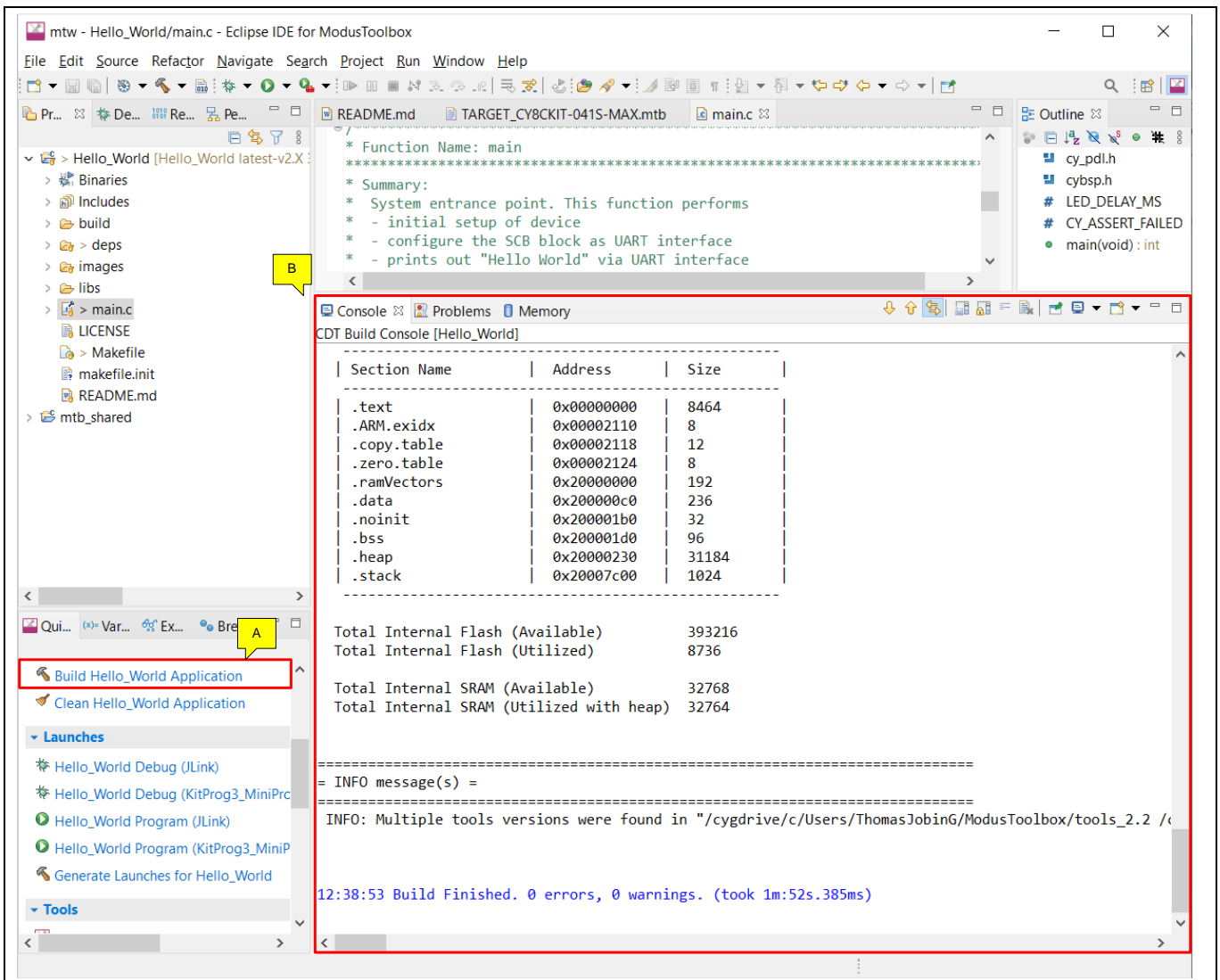


Figure 26 Build the application

If there are any errors, check the steps to make sure you completed the required tasks.

Note: You can also use the CLI to build the application. See **Running ModusToolbox from the Command Line** in the **ModusToolbox™ user guide**. This document is located in the `/ide_2.2/docs/` folder in the ModusToolbox installation directory.

5.8 Part 5: Programming the device

This section shows how to program the PSoC™ 4 device.

ModusToolbox™ uses the SWD protocol to program and debug applications on PSoC™4 devices. For ModusToolbox™ to identify the device on the kit, the kit must be running KitProg3. Some kits are shipped with KitProg2 firmware instead of KitProg3. ModusToolbox™ includes the `fw-loader` command-line tool to switch the KitProg firmware from KitProg2 to KitProg3. See the KitProg Firmware Loader section in the **ModusToolbox™ IDE user guide** for more details.

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

My first PSoC™ 4 design using ModusToolbox™

5.8.1 Program the application

- a) Connect the kit to PC.
- b) Select the application project and click the **<application name> Program (KitProg3_MiniProg4)** shortcut under the **Launches** group in the Quick Panel. 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.

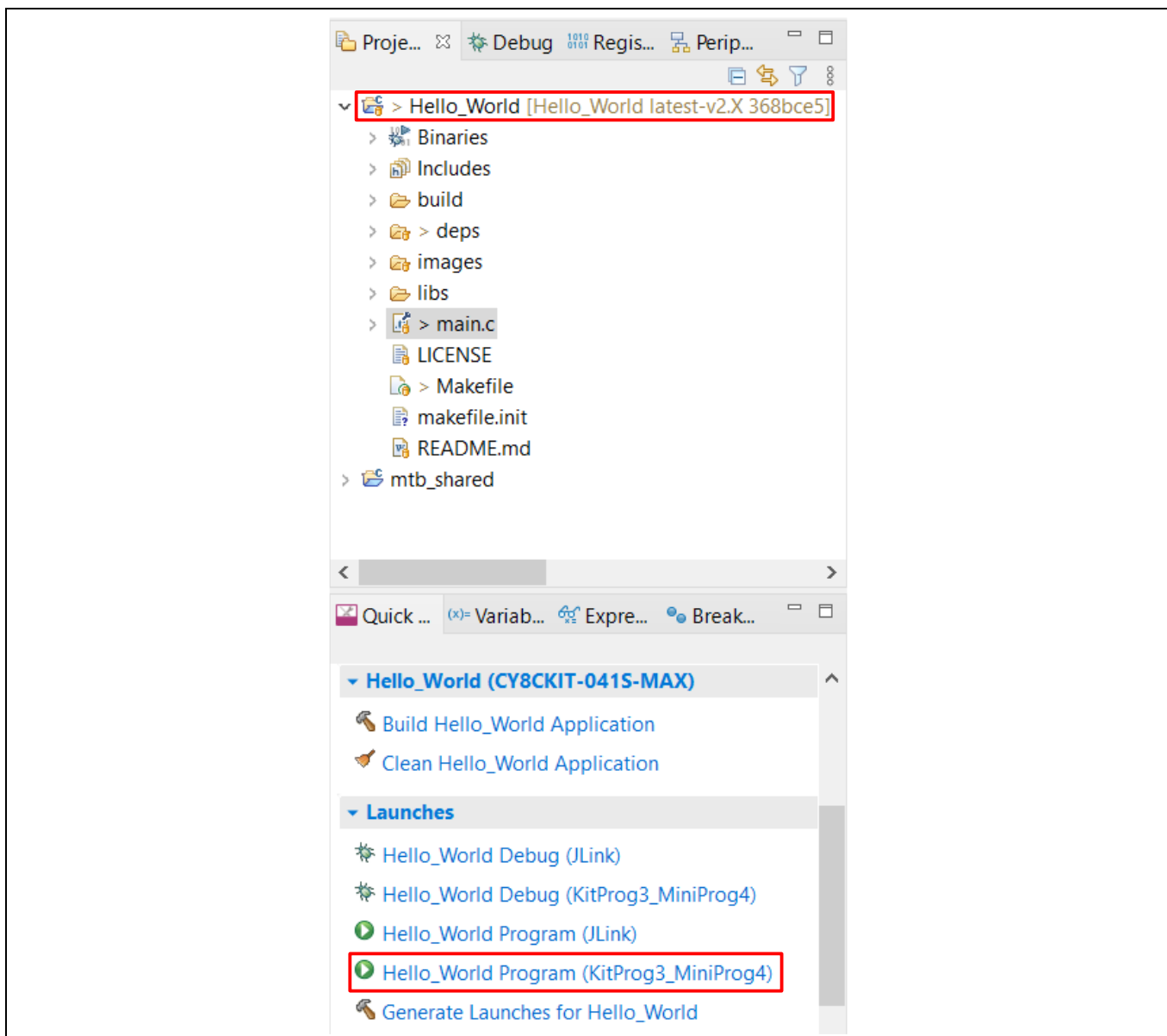
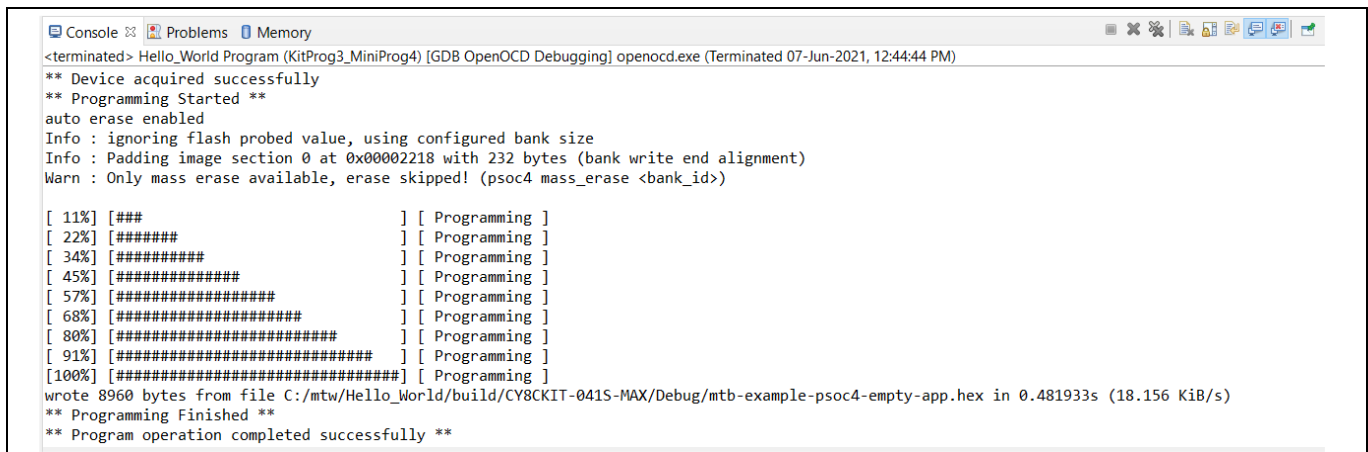


Figure 27 Programming an application to a device

The Console view lists the results of the programming operation.

My first PSoC™ 4 design using ModusToolbox™



```

<terminated> Hello_World Program (KitProg3_MiniProg4) [GDB OpenOCD Debugging] openocd.exe (Terminated 07-Jun-2021, 12:44:44 PM)
** Device acquired successfully
** Programming Started **
auto erase enabled
Info : ignoring flash probed value, using configured bank size
Info : Padding image section 0 at 0x00002218 with 232 bytes (bank write end alignment)
Warn : Only mass erase available, erase skipped! (psoc4 mass_erase <bank_id>)

[ 11%] [###                               ] [ Programming ]
[ 22%] [#####                             ] [ Programming ]
[ 34%] [#####                             ] [ Programming ]
[ 45%] [#####                             ] [ Programming ]
[ 57%] [#####                             ] [ Programming ]
[ 68%] [#####                             ] [ Programming ]
[ 80%] [#####                             ] [ Programming ]
[ 91%] [#####                             ] [ Programming ]
[100%] [#####                             ] [ Programming ]
wrote 8960 bytes from file C:/mtw/Hello_World/build/CY8CKIT-041S-MAX/Debug/mtb-example-psoc4-empty-app.hex in 0.481933s (18.156 KiB/s)
** Programming Finished **
** Program operation completed successfully **
    
```

Figure 28 Console – programming results

5.9 Part 6: Testing your design

This section describes how to test your design.

Follow these steps to observe the output of your design. This application note uses Tera Term as the UART terminal emulator to view the results. You can use any terminal to view the output.

5.9.1 Select the serial port

Launch Tera Term and select the USB-UART COM port as shown in **Figure 29**. Note that your COM port number may be different.

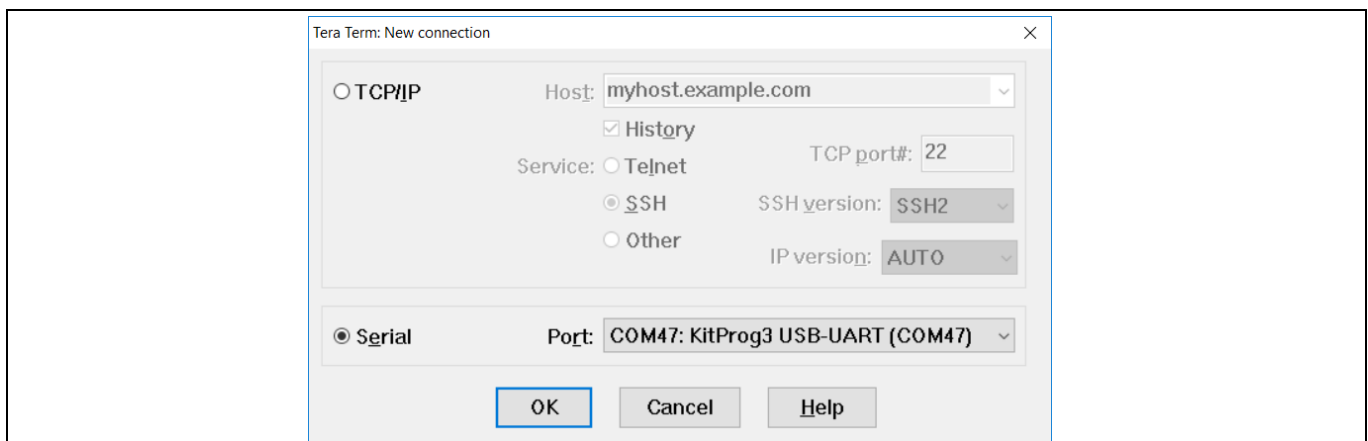


Figure 29 Selecting the KitProg3 COM port in tera term

My first PSoC™ 4 design using ModusToolbox™

5.9.2 Set the baud rate

Go to Setup > Serial port. Set the baud rate to 115200.

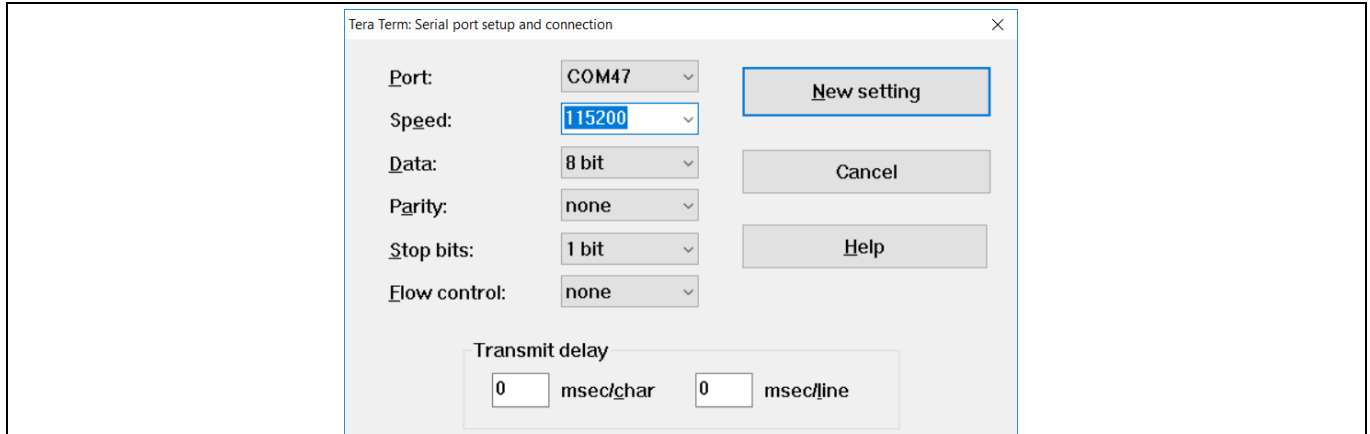


Figure 30 Configuring the baud rate in tera term

5.9.3 Reset the device

Press the reset switch (SW1) on the kit. A “Hello world” message appears on the terminal. The user LED on the kit will start blinking.

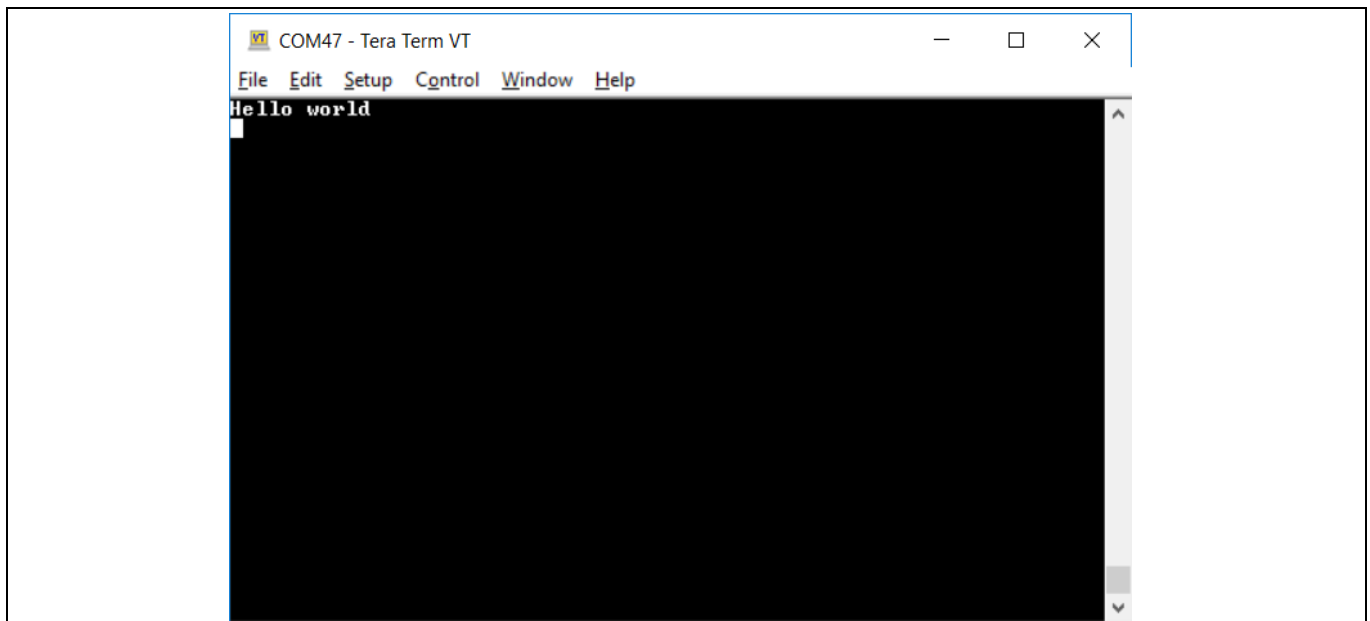


Figure 31 UART message printed from PSoC™ 4

My first PSoC™ 4 design using PSoC™ Creator

6 My first PSoC™ 4 design using PSoC™ Creator

This section:

- Demonstrates how PSoC™ can be programmed to do more than a traditional MCU
- Shows how to build a simple PSoC™ design and install it in a development kit
- Provides detailed steps that make it easy to learn PSoC™ design techniques and how to use [PSoC™ Creator](#)

6.1 Before you begin

6.1.1 Have you installed PSoC™ Creator?

Download and install PSoC™ Creator from the [PSoC™ Creator home page](#). Note that the installation of the toolset may take a long time – see the PSoC™ Creator Release Notes for more information.

6.1.2 Do you have a development kit or prototyping kit?

Testing this design requires one of the kits listed in [Table 9](#), which has an integrated programmer.

Table 9 List of PSoC™ 4 pioneer kits, prototyping kits, and supported Devices

Kit name	Kit type	Supported device family	Part number
CY8CKIT-040	Pioneer kit	PSoC™ 4000	CY8C4014LQI-422
CY8CKIT-041	Pioneer kit	PSoC™ 4100S	CY8C4146AZI-S433
CY8CKIT-042	Pioneer kit	PSoC™ 4200	CY8C4245AXI-483
CY8CKIT-044	Pioneer kit	PSoC™ 4200M	CY8C4247AZI-M485
CY8CKIT-046	Pioneer kit	PSoC™ 4200L	CY8C4248BZI-L489
CY8CKIT-042-BLE	Pioneer kit	PSoC™ 4200 BLE	CY8C4247LQI-BL483
CY8CKIT-045S	Pioneer kit	PSoC™ 4500S	CY8C4548AZI-S485
CY8CKIT-043	Prototyping kit	PSoC™ 4200M	CY8C4247AZI-M485
CY8CKIT-145	Prototyping kit	PSoC™ 4000S	CY8C4045AZI-S413
CY8CKIT-147	Prototyping kit	PSoC™ 4100PS	CY8C4145LQI-PS433
CY8CKIT-149	Prototyping kit	PSoC™ 4100S Plus	CY8C4147AZI-S475

6.1.3 Want to see the project in action?

If you do not want to go through the design process, you can get the completed PSoC™ Creator project using **Find Code Example** in PSoC™ Creator (**File > Code Example... > CE230991_My_First_Project**); see [Code examples](#) for more details. You can then jump to the [Build](#) and [Program](#) steps.

6.2 About the design

This design simply blinks two LEDs using a TCPWM Component, as shown in [Figure 32](#). The TCPWM is configured in PWM mode. The two complementary outputs of this PWM control the LEDs. The PWM operates at a very low frequency and 50 percent duty cycle so that the toggling of the LEDs is visible. If you use a dual-color LED instead of two separate LEDs, this project can toggle the color of the dual-color LED.

My first PSoC™ 4 design using PSoC™ Creator

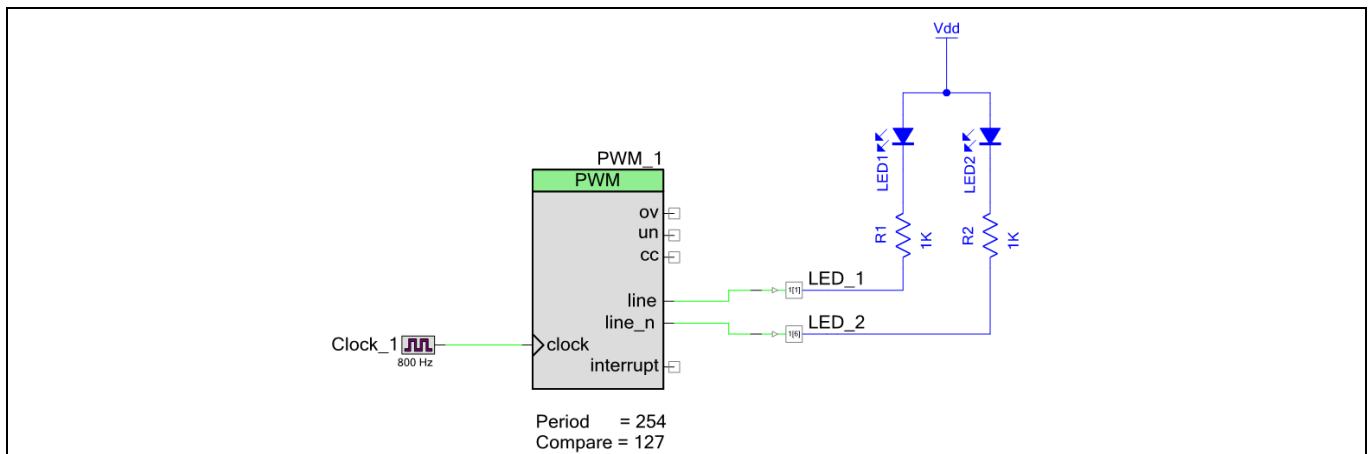


Figure 32 My first PSoC™ 4 design

6.3 Part 1: Create the design

This section takes you on a step-by-step guided tour of the design process. It starts with creating an empty project and guides you through hardware and firmware design entry.

1. Start PSoC™ Creator, and from the **File** menu choose **New > Project**, as shown in **Figure 33**.

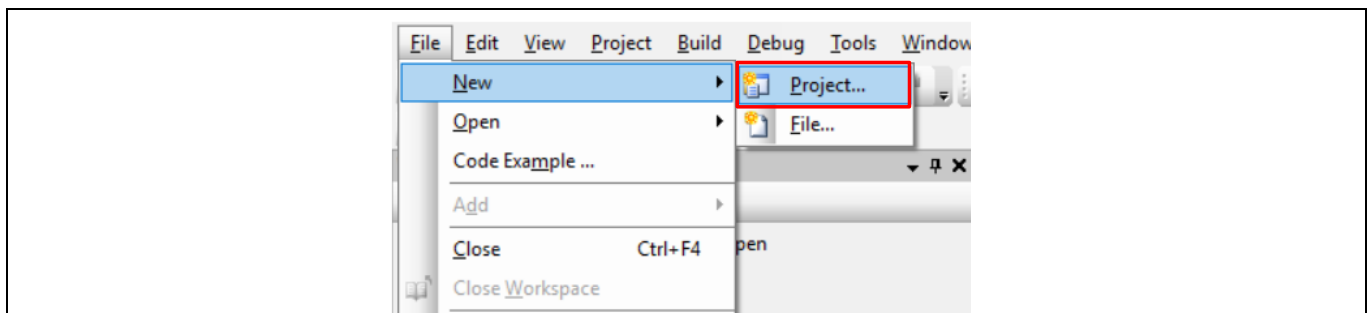


Figure 33 Creating a new project

2. Select your development kit in the pop-up window. For example, if you have a CY8CKIT-149, select **Kit: CY8CKIT-149 (PSoC™ 4100S Plus)** and click **Next**. If you do not see your PSoC™ 4 development kit listed in the menu, download and install the kit setup for your kit from the website.

Alternately, you can also select the target device radio button instead of the target hardware and select the appropriate device and click **Next**.

My first PSoC™ 4 design using PSoC™ Creator

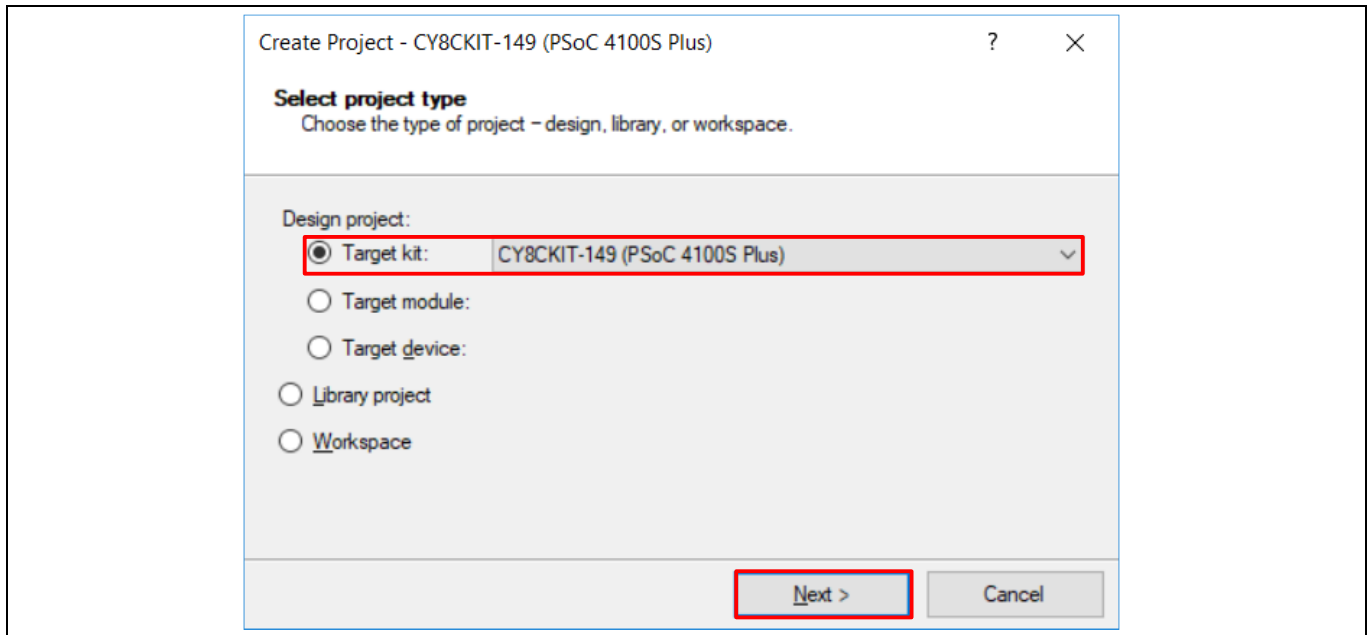


Figure 34 Create a new empty PSoC™ 4 project

3. Select the option **Empty Schematic** from the next window and click **Next**.

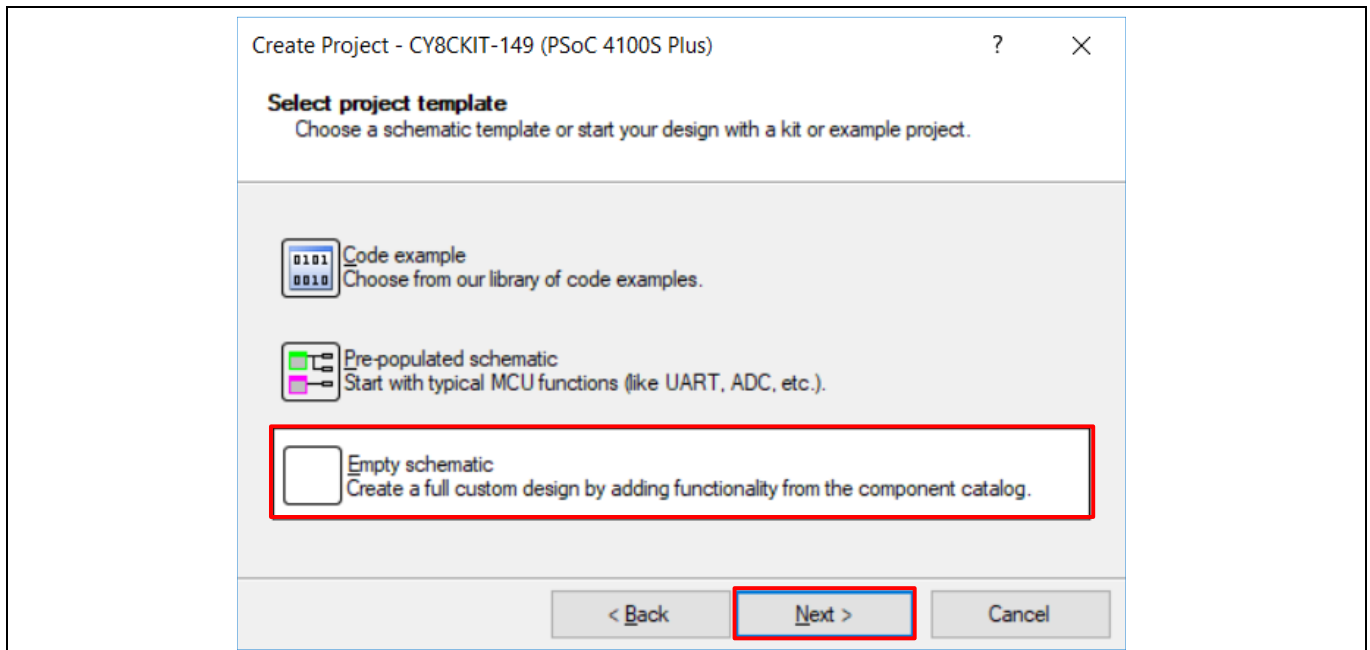


Figure 35 Select empty schematic

4. Provide a project name (for example, “My_First_Project”) and Workspace Name as shown in **Figure 36**. Choose an appropriate location for your new project, and click **Finish**.

My first PSoC™ 4 design using PSoC™ Creator

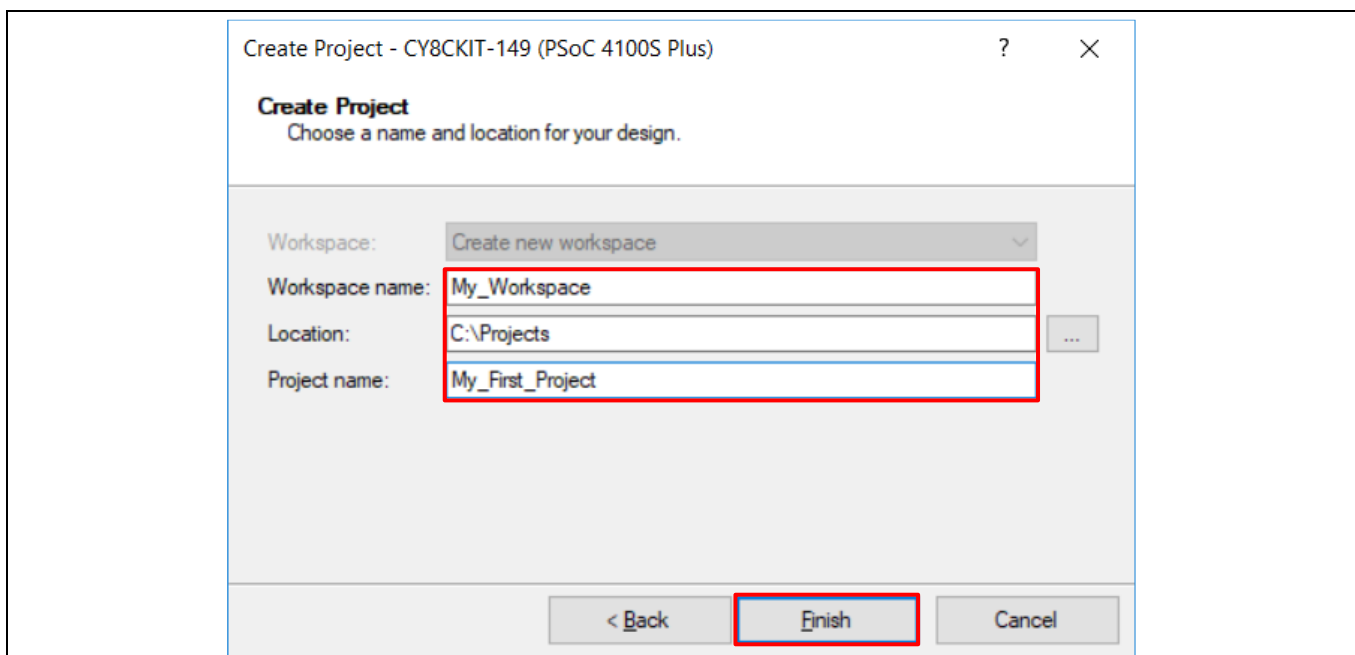


Figure 36 Selecting project name and location

5. Creating a new project generates a project folder with a baseline set of files shown in the **Workspace Explorer** (see [Figure 37](#)). To open the project schematic file, double-click *TopDesign.cysch*.

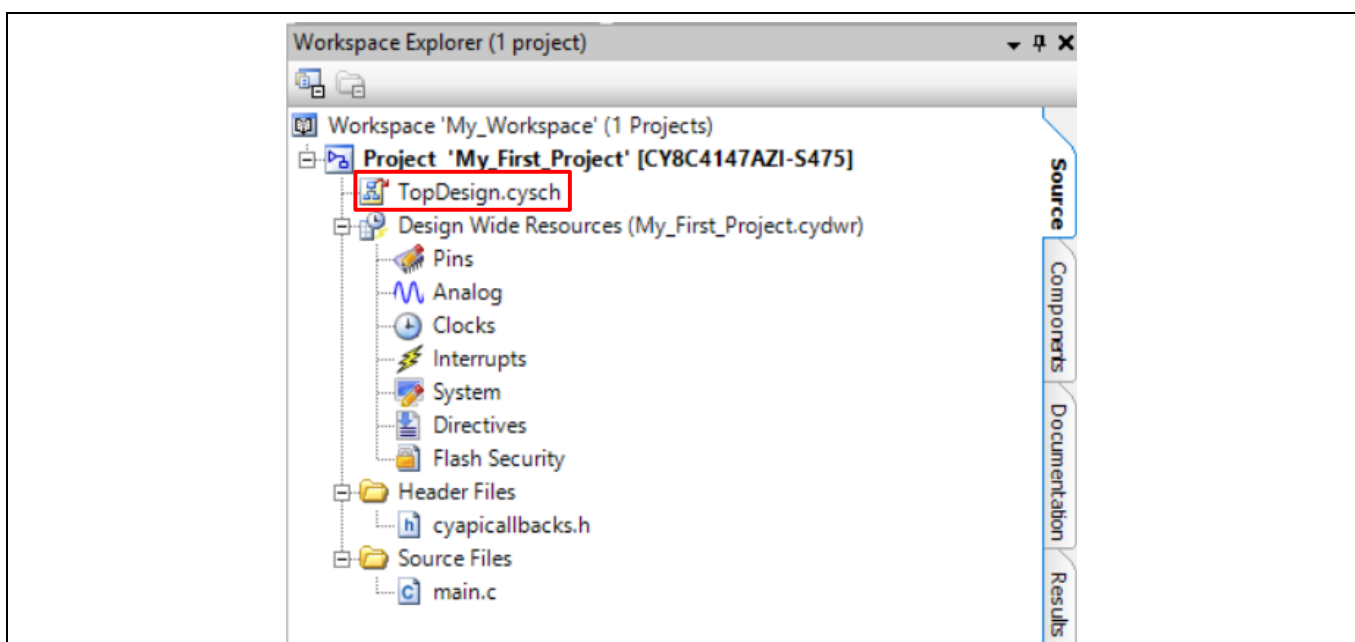


Figure 37 Opening TopDesign schematic

6. Drag one **PWM (TCPWM mode)** Component from the **Component Catalog** onto the schematic, as shown in [Figure 38](#).

My first PSoC™ 4 design using PSoC™ Creator

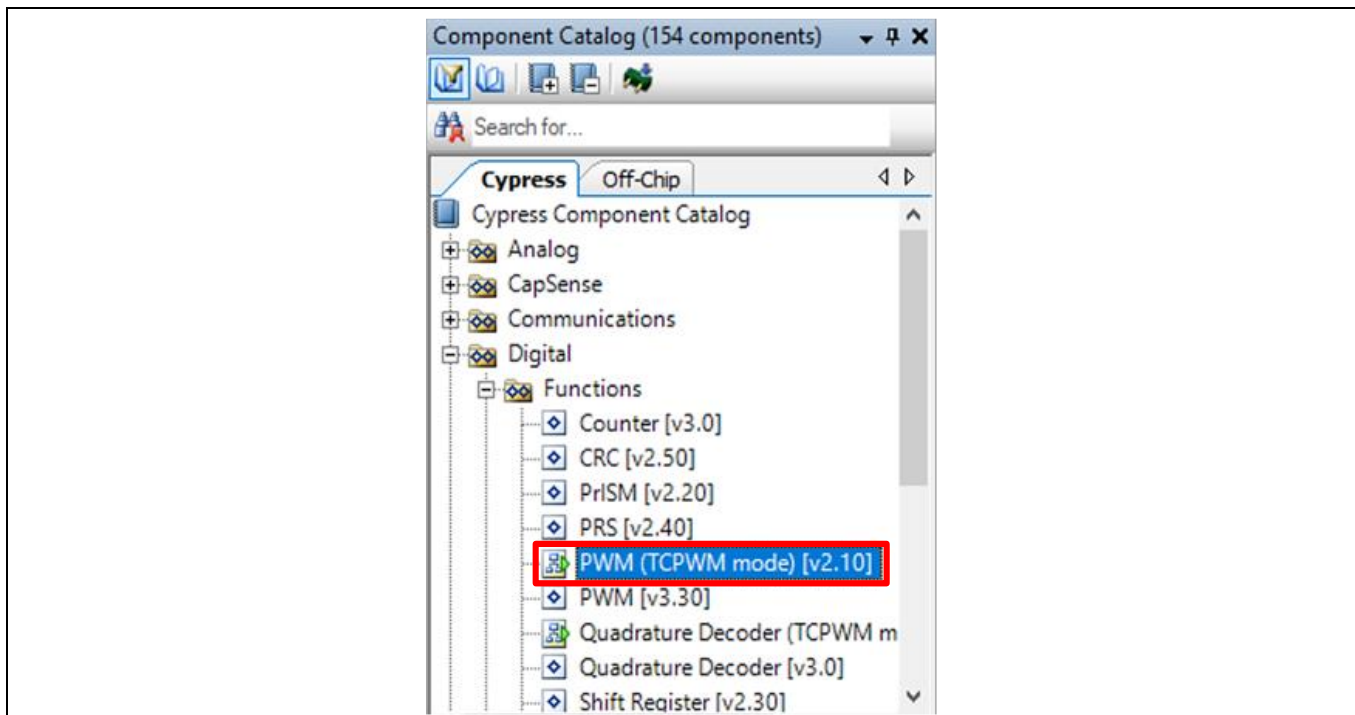


Figure 38 Location of the PWM Component

7. Double-click the PWM Component on the schematic to configure the Component properties, as shown in **Figure 39**. Click the **PWM** tab, and set the Period value to 254 and the Compare value to 127 to generate a PWM signal with a 50 percent duty cycle.

Set the **Prescaler** to 8x, to divide the input clock frequency by 8.

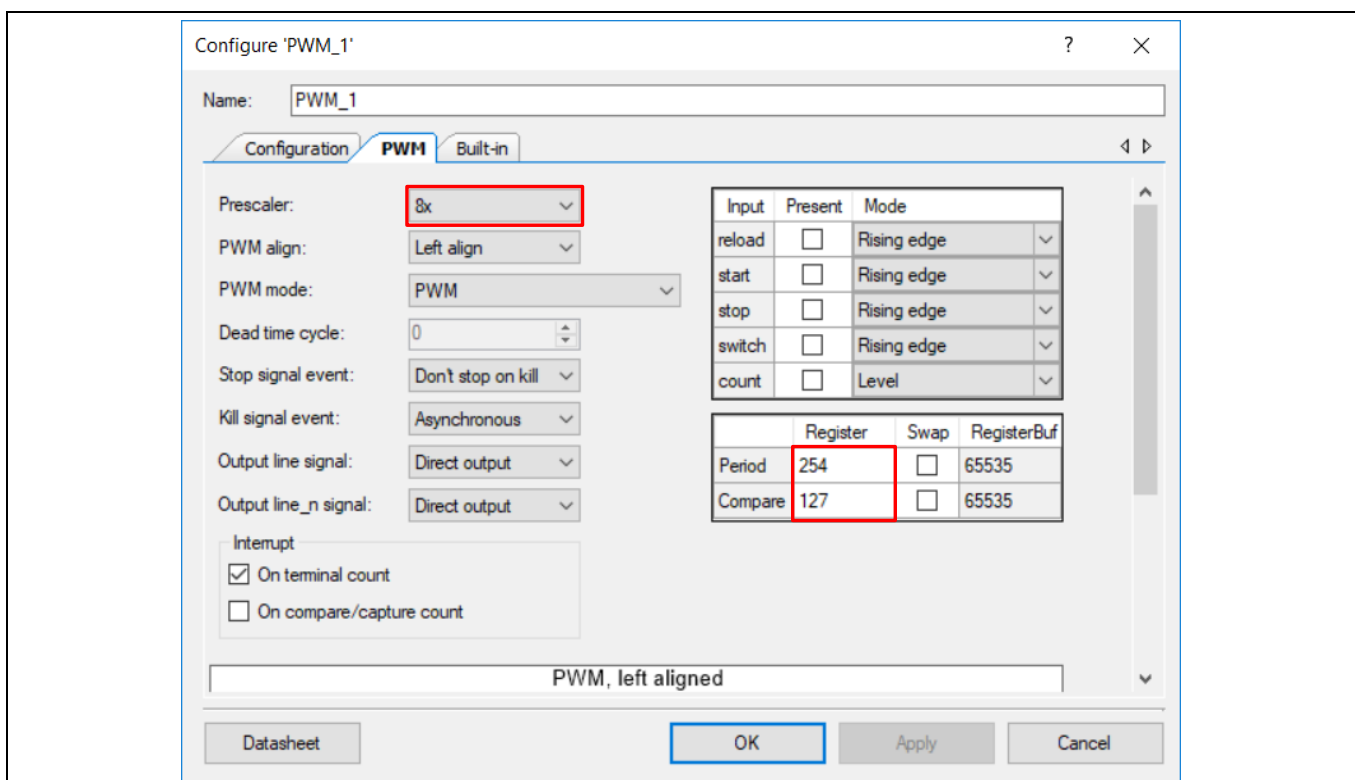


Figure 39 Configuring the PWM Component

My first PSoC™ 4 design using PSoC™ Creator

- 8. A PWM Component requires an input clock for its operation. Drag and drop a **Clock** Component onto the schematic, and configure the **Frequency** to 800 Hz by double-clicking on the Component, as shown **Figure 40** and **Figure 41**. Since the Prescaler value set in PWM Component is 8, the effective input clock of the PWM is only 100 Hz. Therefore, the PWM period of 254 results in a PWM output time period of 2.54 seconds.

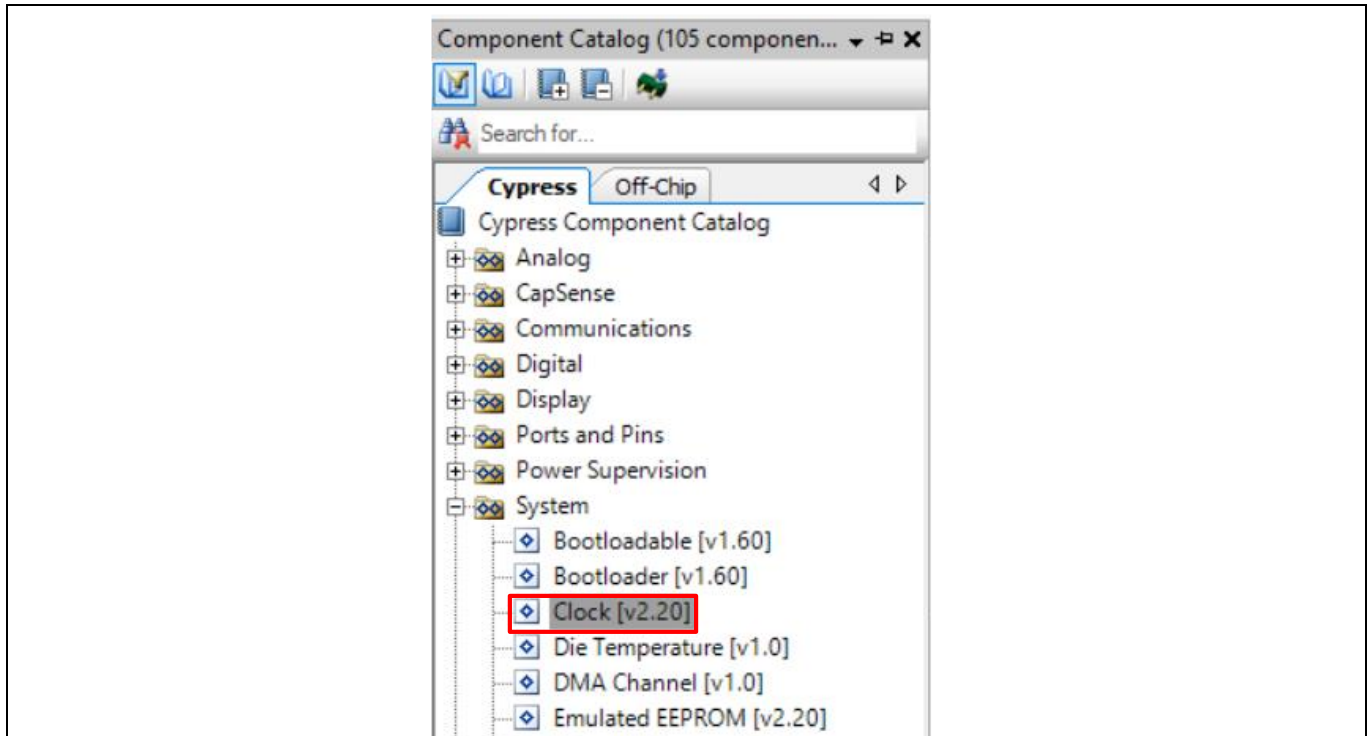


Figure 40 Location of the clock Component

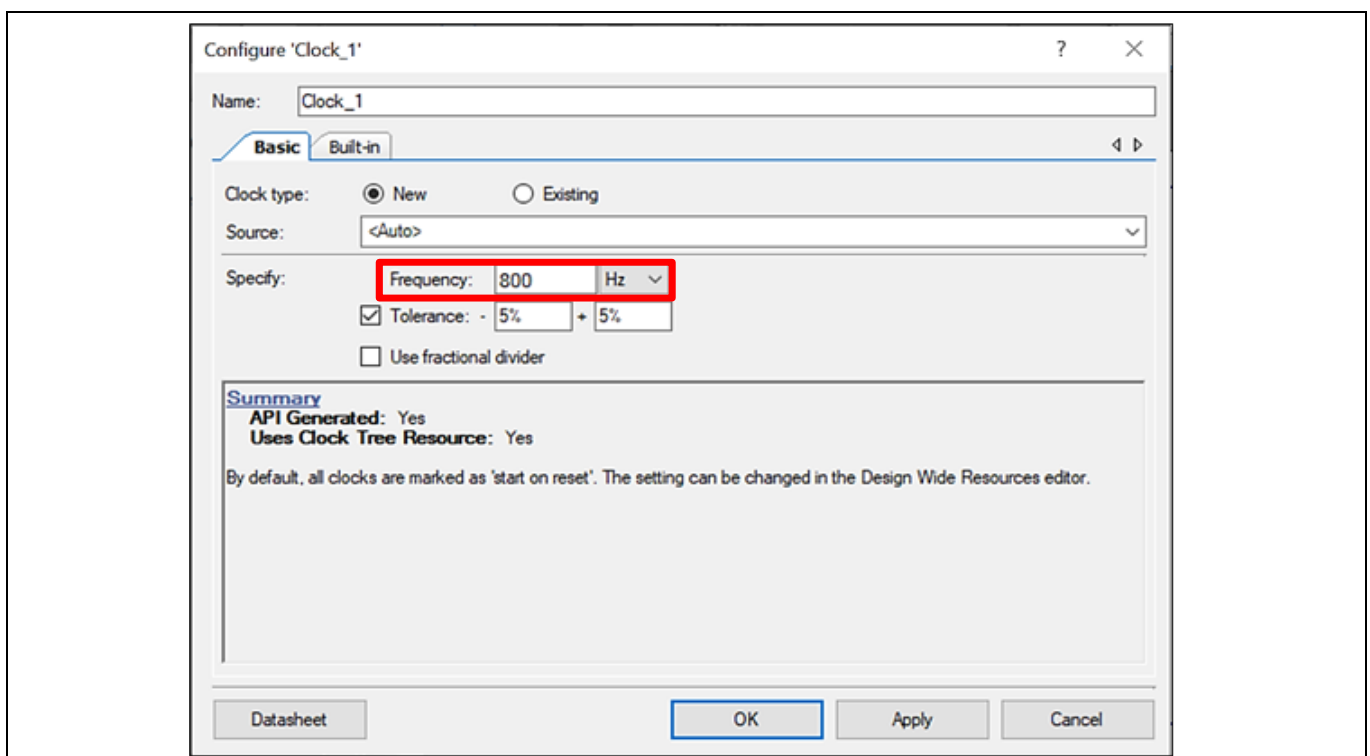


Figure 41 Configuring the clock Component

My first PSoC™ 4 design using PSoC™ Creator

9. Drag and drop a **Digital Output Pin** Component. Change the name to LED_1 as shown in **Figure 42** and **Figure 43**. Add another Digital Output Pin Component and change its name to LED_2.

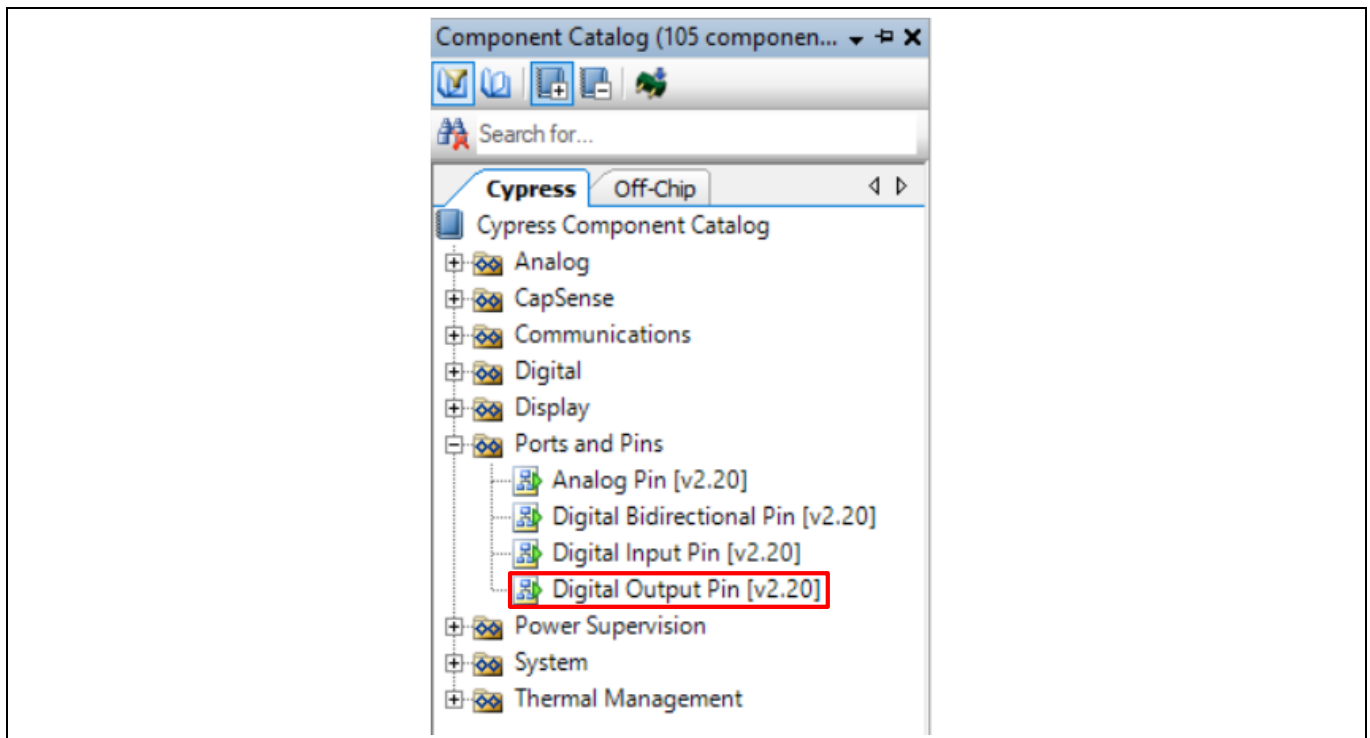


Figure 42 Location of the digital output pin Component

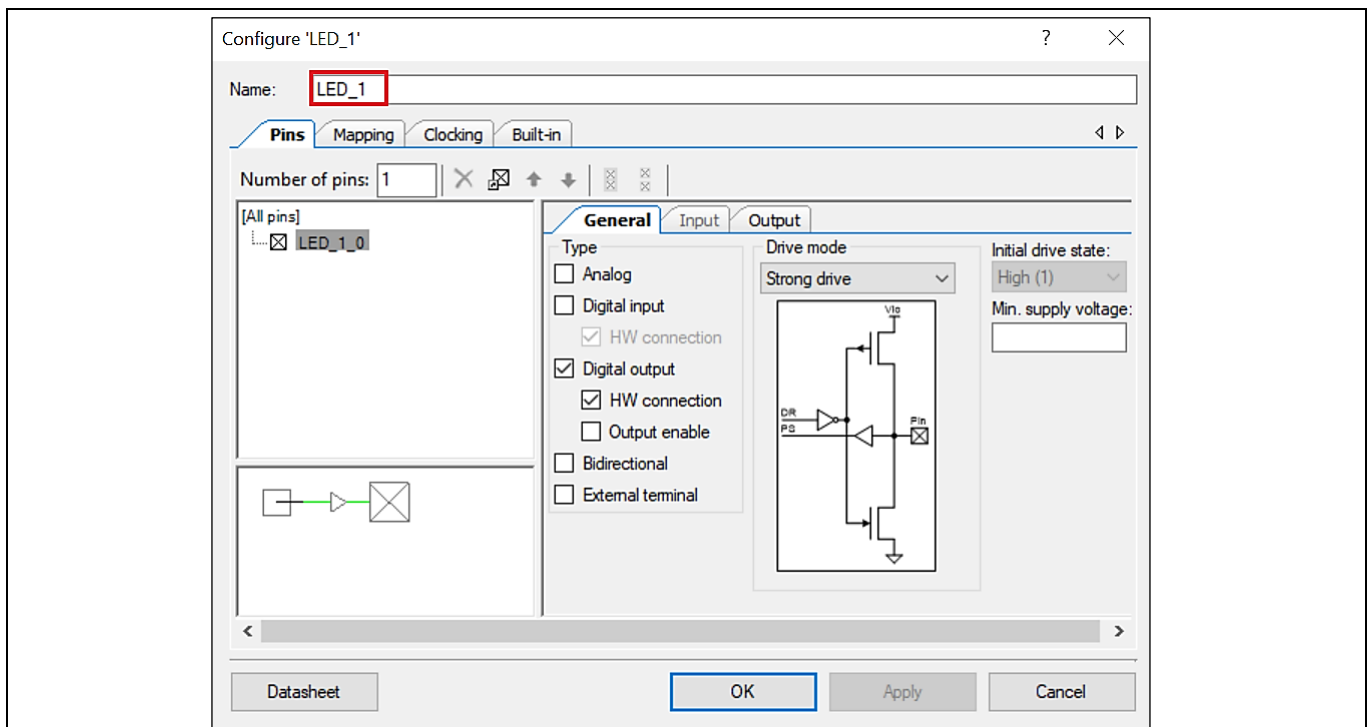


Figure 43 Renaming a pin Component

My first PSoC™ 4 design using PSoC™ Creator

10. In the schematic window, select the wire tool as shown in **Figure 44**, or press **W**.

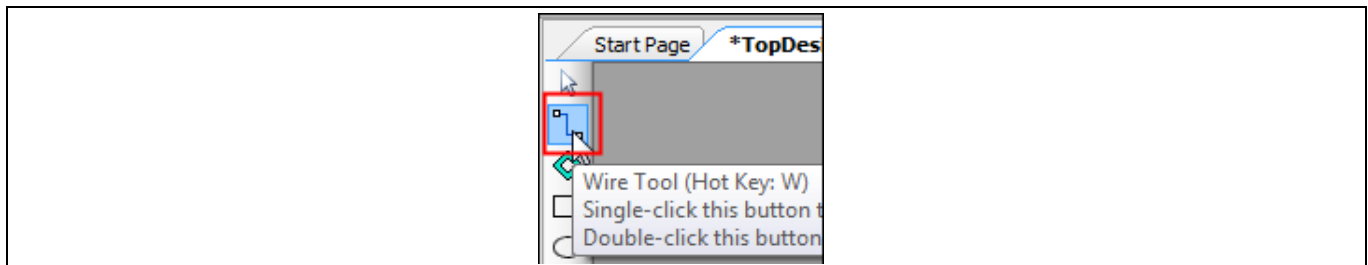


Figure 44 Selecting the wire tool

11. Wire the Components together, as shown in **Figure 45**.

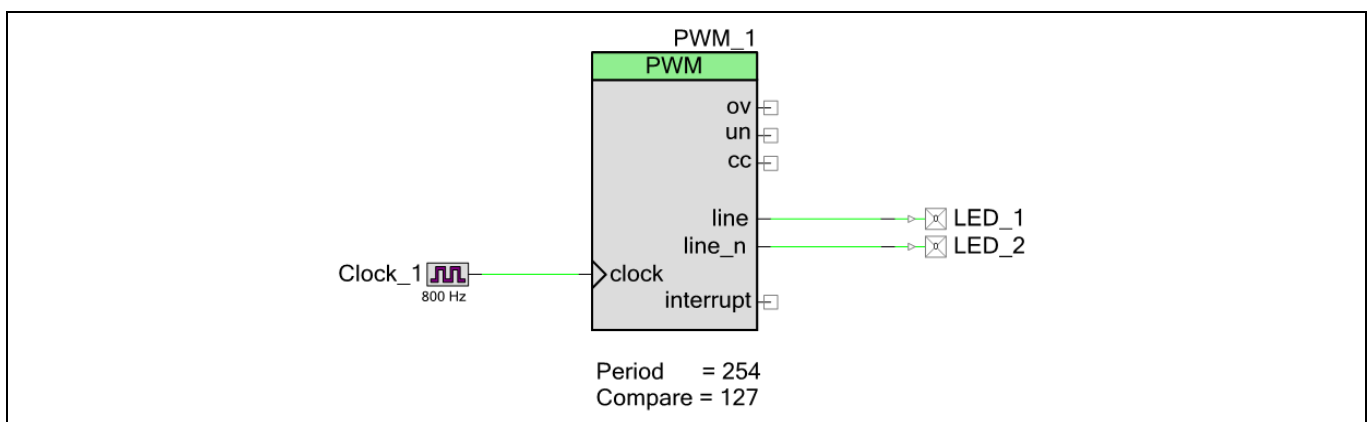


Figure 45 Wiring the schematic

12. Most Components are disabled at device reset (the major exception being the Clock Component, which is automatically started as a default), and you must add code to the project to enable them. Open *main.c* from **Workspace Explorer** and add code to the `main()` function, as provided in **Code Listing 2**.

Code Listing 2 Enabling the PWM Component

```
int main(void)
{
    /* Enable and start the PWM */
    PWM_1_Start();

    for(;;)
    {

    }
}
```

13. Select **Build My_First_Project** from the Build menu. Notice in the **Workspace Explorer** window that PSoC™ Creator automatically generates source code files for the PWM, Clock, and Digital Output Pin Components, as shown in **Figure 46**.

My first PSoC™ 4 design using PSoC™ Creator

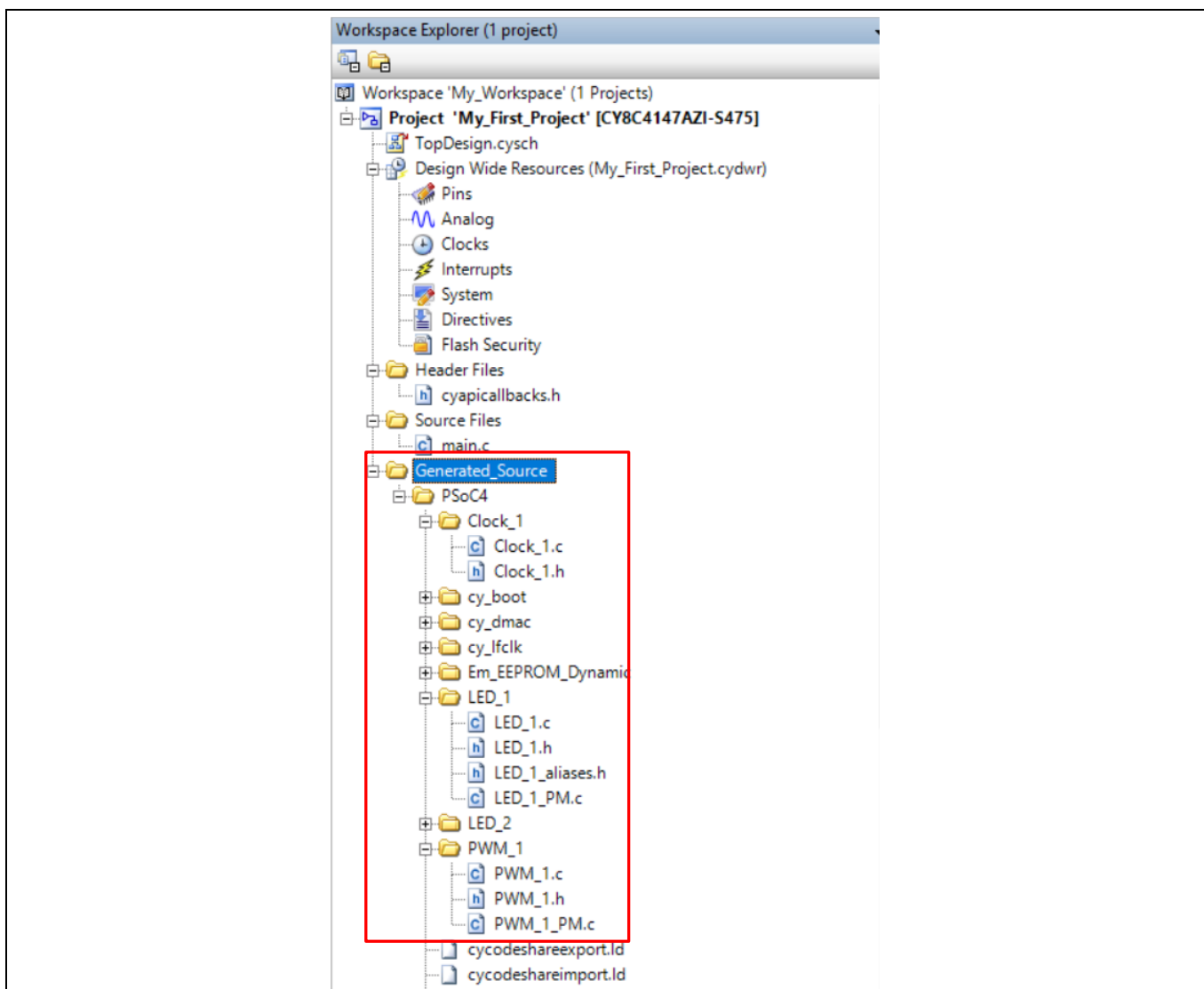


Figure 46 Generated source files

14. Open the file *My_First_Project.cydwr* (Design-Wide Resource file) from **Workspace Explorer** and click the **Pins** tab. You can use this tab to select the device pins for the outputs LED_1 and LED_2.

Figure 47 shows the pin configuration to connect the LED_1 and LED_2 pins to the LEDs in the **CY8CKIT-149 PSoC™ 4 prototyping kit**. See **Table 10** if you are using a different PSoC™ 4 Pioneer Kit, or **Table 11** if you are using a PSoC™ 4 prototyping kit.

My first PSoC™ 4 design using PSoC™ Creator

6.4 Part 2: Program the device

This section shows how to program the device. Connect the kit board to your computer using the USB cable.

1. Select the PSoC™ Creator menu item **Debug > Select Debug Target**, as shown in [Figure 48](#).

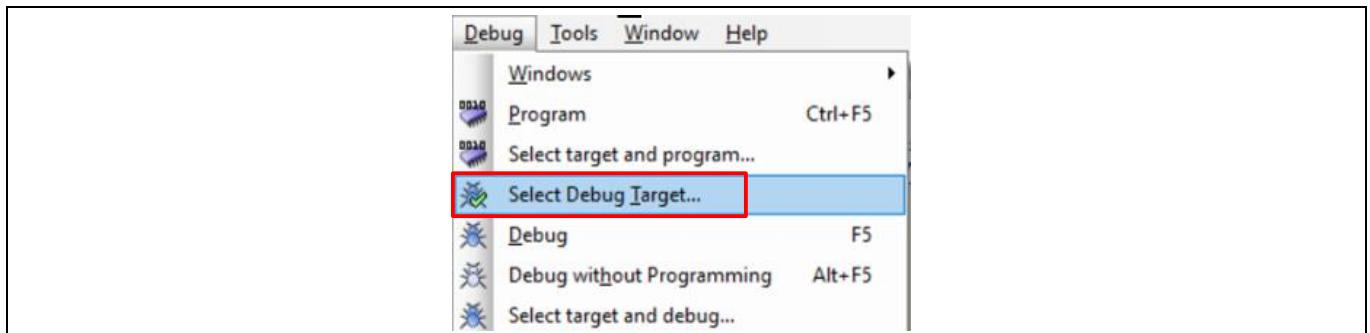


Figure 48 Selecting debug target

2. In the **Select Debug Target** dialog box, click **Port Acquire**, and then click **Connect**, as shown in [Figure 49](#). Click **OK** to close the dialog box.

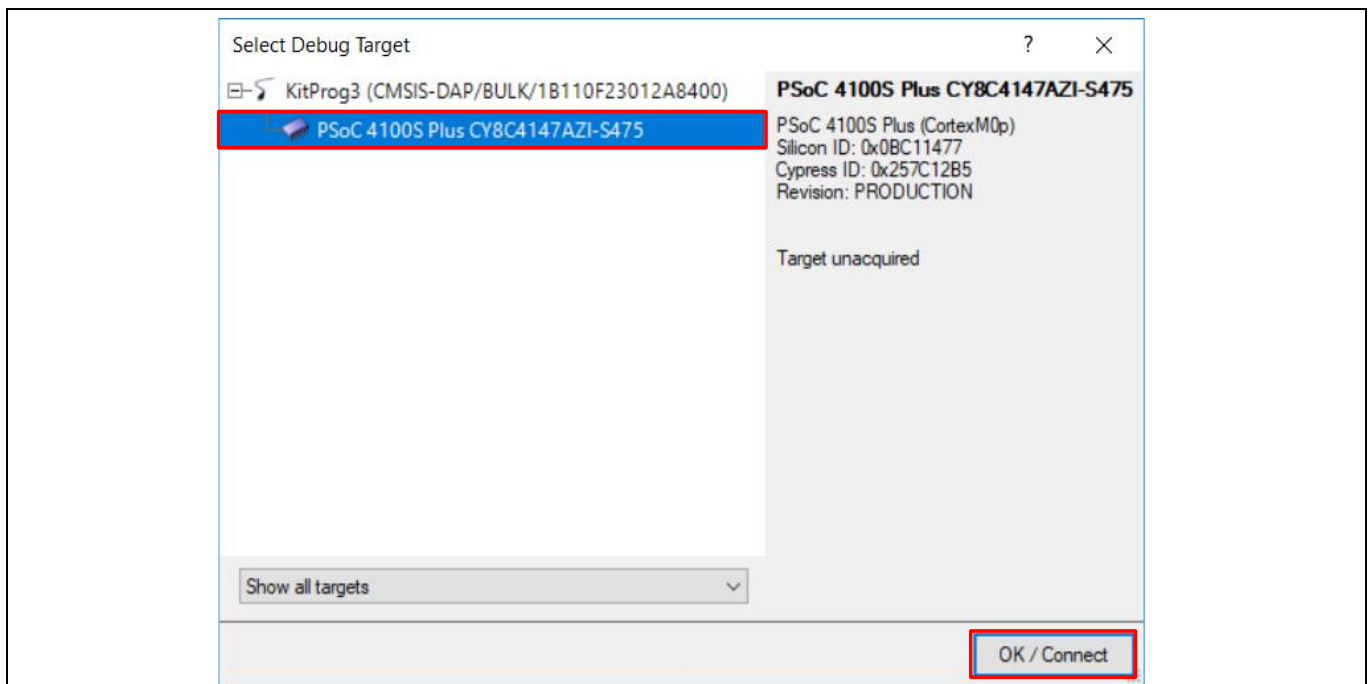


Figure 49 Connecting to a device

3. Choose the menu item **Debug > Program** to program the device with the project, as shown in [Figure 50](#).

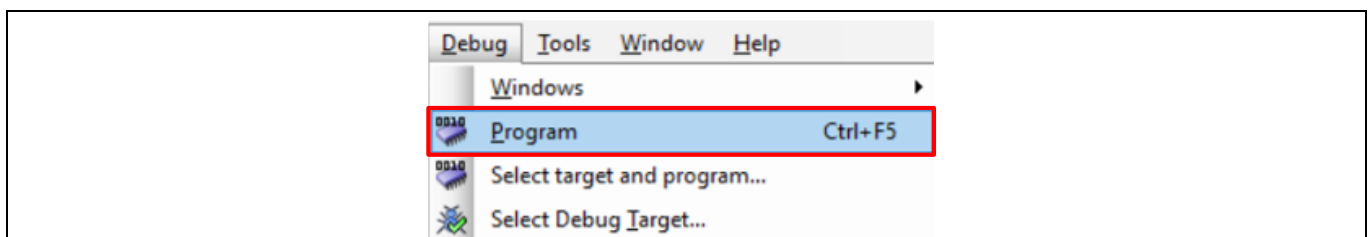


Figure 50 Programming the device

My first PSoC™ 4 design using PSoC™ Creator

4. You can view the programming status on the status bar (lower-left corner of the window), as shown in [Figure 51](#).

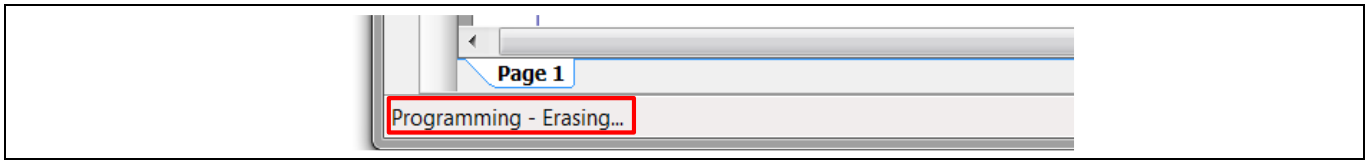


Figure 51 Programming status

5. After the device is programmed, verify the operation of the project by viewing the toggling of the LEDs.

Summary

7 Summary

This application note explored the PSoC™ 4 architecture and development tools. PSoC™ 4 is a truly programmable embedded system-on-chip, integrating configurable analog and digital peripheral functions, memory, and an Arm® Cortex®-M0/M0+ microcontroller on a single chip. Because of the integrated features and low-leakage power modes, PSoC™ 4 is an ideal choice for low-power and cost-effective embedded systems.

This application note also guided you to a comprehensive collection of resources to accelerate in-depth learning about PSoC™ 4.

References

References

- [1] [AN54181](#): Getting started with PSoC™ 3
- [2] [AN77759](#): Getting started with PSoC™ 5LP

Revision history

Revision history

Document version	Date of release	Description of changes
**	2013-01-24	New Application Note
*A	2013-04-11	Demo project changed to leverage Pioneer kit. Added architecture introduction.
*B	2013-05-09	Reformatted graphics. Updated links.
*C	2013-12-19	Updated attached Associated Project files. Updated content across the entire document. Updated in new template.
*G	2014-04-10	Updated the projects and the respective section in the AN to support PSoC. Creator 3.0 SP1 and PSoC 4000 device.
*H	2014-09-25	Added Code Examples section. Minor edits and format changes throughout the document.
*I	2015-03-17	Added More Information section. Removed detailed feature descriptions. Updated for PSoC 4200M family of devices.
*J	2015-09-10	Updated PSoC™ resources and PSoC™ is more than an MCU . Added the following sections: Convert Project to Bootloadable for CY8CKIT-049, Bootload Your CY8CKIT-049, and More PSoC 4 Code Examples. Updated Figure 9 .
*K	2015-09-16	Updated for PSoC 4200L. Updated the example projects to PSoC Creator 3.3.
*L	2015-12-30	Updated the example projects to PSoC Creator 3.3 SP1.
*M	2016-02-05	Updated for PSoC 4000S and PSoC 4100S. Updated the example projects to PSoC Creator 3.3 SP2. Updated Table 9 , Table 10 to add PSoC 4200 BLE.
*N	2017-04-19	Updated logo and copyright
*O	2017-10-06	Added references to PSoC 4100S Plus throughout the document. Updated Table 1 to add PSoC 4100S Plus. Updated PSoC™ resources with the reference of AN64846 - Getting Started with CAPSENSE™ .
*P	2017-11-03	Updated the example projects to PSoC Creator 4.2 Updated Table 1 to add supported kit Updated Table 9 to add CY8CKIT-49, CY8CKIT-145 and CY8CKIT-149 Added Table 11 to include CY8CKIT-145 and CY8CKIT-149 pin mapping Updated Figure 7 . Added example project of CY8CKIT-145 and CY8CKIT-149 as a part of the AN79953.zip file in this application note landing page.
*Q	2018-03-06	Updated template Updated for PSoC 4100PS Minor edits and format changes throughout the document

Revision history

Document version	Date of release	Description of changes
		Added example project of CY8CKIT-147 as a part of the AN79953.zip file
*R	2018-05-04	Updated template Corrected the link to PSoC 4100S Plus in PSoC™ resources
*S	2019-09-06	Updated Table 1 to add PSoC 4500 and update the specifications of PSoC 4100S Plus device family. Added Motor Control Accelerator feature.
*T	2020-11-09	Updated for ModusToolbox support for some of the PSoC 4 devices. Corrected the links in PSoC™ resources . PSoC 4 families are recategorized in PSoC™ 4 feature set Added new section - My first PSoC™ 4 design using ModusToolbox™ Removed sections : Convert Project to Bootloadable for CY8CKIT-049, Bootload Your CY8CKIT-049, and More PSoC 4 Code Examples
*U	2021-07-08	Updated to Infineon template Updated Figure 2 , Figure 3 , Figure 4 , and Figure 5 to the latest MTB 2.3 version Updated document with PSoC™ 4100S Max device Changed the default device to PSoC™ 4100S Max and updated Figure 9 Updated Modify the design section

Trademarks

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

Edition 2021-07-08

Published by

Infineon Technologies AG

81726 Munich, Germany

© 2021 Infineon Technologies AG.

All Rights Reserved.

Do you have a question about this document?

Go to www.cypress.com/support

Document reference

001-79953 Rev. *U

IMPORTANT NOTICE

The information contained in this application note is given as a hint for the implementation of the product only and shall in no event be regarded as a description or warranty of a certain functionality, condition or quality of the product. Before implementation of the product, the recipient of this application note must verify any function and other technical information given herein in the real application. Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this application note.

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

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

WARNINGS

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

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