Getting Started with PSoC 6 MCU on PSoC Creator

Associated Part Family

All **PSoC® 6 MCU** devices

Software Version

PSoC Creator™ 4.2

About this document

Scope and purpose

AN221774 introduces the PSoC 6 MCU, a dual-CPU programmable system-on-chip with Arm® Cortex®-M4 and Cortex-M0+ processors. This application note helps you explore PSoC 6 MCU architecture and development tools, and shows you how to create your first project using PSoC Creator. This application note also guides you to more resources available online to accelerate your learning about PSoC 6 MCU. To get started with the PSoC 6 MCU with Bluetooth® Low Energy (LE) Connectivity device family, refer to **AN210781** – Getting Started with PSoC 6 MCU with BLE Connectivity on PSoC Creator.

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**. Please visit the **GitHub** site for a comprehensive collection of code examples using ModusToolbox IDE for PSoC 6. You can also explore the PSoC video library **here**.

Table of contents

1 Introduction

PSoC 6 MCU is an ultra-low-power PSoC device with a dual-CPU architecture tailored for smart homes, IoT gateways, and so on. The PSoC 6 MCU device is a programmable embedded system-on-chip that integrates the following features on a single chip:

- Single-CPU microcontroller: Arm Cortex-M4 (CM4) or Dual-CPU microcontroller: Arm Cortex-M4 (CM4) and Cortex-M0+ (CM0+).
- Programmable analog and digital peripherals.
- Up to 2 MB of flash and 1 MB of SRAM.
- Fourth-generation CapSense® technology.
- PSoC 6 MCU is suitable for a variety of power-sensitive applications such as the following:
  - Smart home sensors and controllers.
  - Smart home appliances.
  - Gaming controllers.
  - Sports, smart phone, and virtual reality (VR) accessories.
  - Industrial sensor nodes.
  - Industrial logic controllers.
  - Advanced remote controllers.

The programmable analog and digital subsystems allow flexibility and dynamic fine-tuning of the design using **PSoC Creator**, the schematic-based design tool.

Figure 1 illustrates an application-level block diagram for a real-world use case using PSoC 6 MCU.

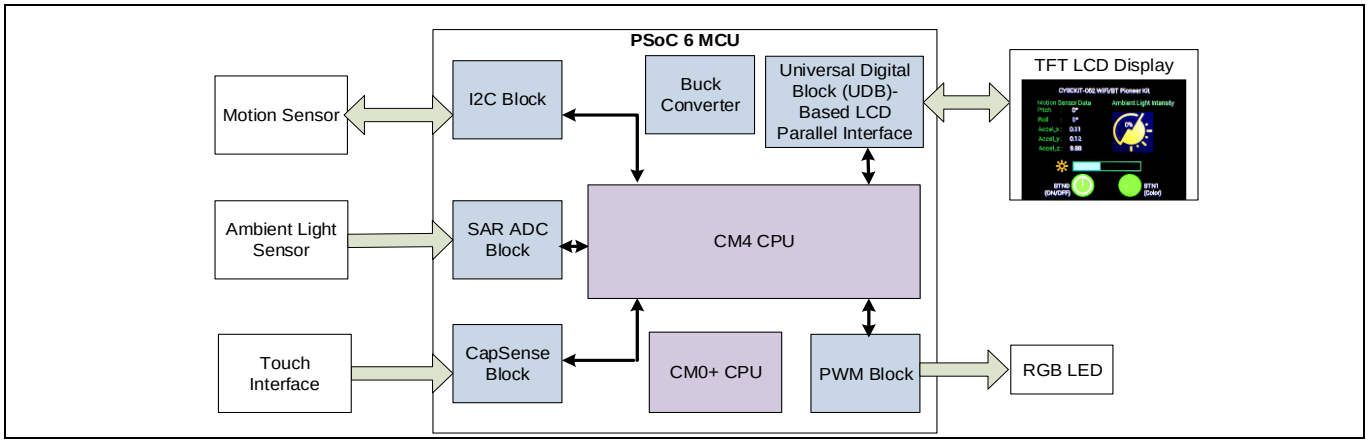

Figure 1 Application-Level Block Diagram Using PSoC 6 MCU

PSoC 6 MCU is a highly capable and flexible solution. For example, the real-world use case in **Figure 1** takes advantage of these features:

- A buck converter for ultra-low-power operation.
- An analog front end (AFE) within the device to condition and measure sensor outputs such as ambient light sensor.
- Serial Communication Blocks (SCBs) to interface with multiple digital sensors such as motion sensors.
- CapSense technology for reliable touch and proximity sensing.

- Digital logic (Universal Digital Blocks or UDBs) and peripherals (Timer Counter PWM or TCPWM) to drive the display and LEDs respectively.
- Product security features managed by CM0+ CPU and application features executed by CM4 CPU.

See **Device Features** and the device datasheets for more details.

This application note introduces you to the capabilities of PSoC 6 MCU, gives an overview of the development ecosystem, and gets you started with a simple design wherein you learn to use PSoC 6 MCU. This design is available as code example **CE221773** for PSoC Creator.

For hardware design considerations, see **AN218241 – PSoC 6 MCU Hardware Design Considerations**.

1.1 Prerequisites

Before you get started, make sure that you have a development kit and have installed the required software. It is recommended that you download the code example for reference.

1.1.1 Hardware

- **CY8CKIT-062-WiFi-BT PSoC 6 Wi-Fi-BT Pioneer Kit**
- **CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit** or
- **CY8CPROTO-062-4343W PSoC 6 Wi-Fi BT Prototyping Kit** (not supported in PSoC Creator).

1.1.2 Software

- **PSoC Creator 4.2** with **Peripheral Driver Library** (PDL v3.1.x or later)
- **CE221773** – PSoC 6 MCU Hello World Example

2 Development Ecosystem

2.1 PSoC Resources

A wealth of data available **here** helps you to select the right PSoC device and quickly and effectively integrate it into your design. For a comprehensive list of PSoC 6 MCU resources, see **How to Design with PSoC 6 MCU - KBA223067**. The following is an abbreviated list of resources for PSoC 6 MCU.

- **Overview: PSoC Portfolio**, PSoC Roadmap
- **Product Selectors: PSoC 6 MCU**
- **Datasheets** describe and provide electrical specifications for each device family.
- **Application Notes and Code Examples** cover a broad range of topics, from basic to advanced level. You can also browse our collection of code examples. See **Code Examples**.
- **Technical Reference Manuals (TRMs)** provide detailed descriptions of the architecture and registers in each device family.
- **PSoC 6 MCU Programming Specification** provides the information necessary to program the nonvolatile memory of PSoC 6 MCU devices.
- **CapSense Design Guides**: Learn how to design capacitive touch-sensing applications with PSoC devices.
- Development Tools
  - **CY8CKIT-062-WiFi-BT PSoC 6 Wi-Fi-BT Pioneer Kit** is a development kit that supports the PSoC 62 series MCU along with Wi-Fi and BT connectivity.
  - **CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit** is an easy-to-use and inexpensive development platform for PSoC 63 series MCU with BLE Connectivity.
  - **CY8CPROTO-062-4343W PSoC 6 Wi-Fi BT Prototyping Kit** is a development kit that supports the PSoC 62 series MCU along with CYW4343W module-based Wi-Fi and Bluetooth connectivity for development on ModusToolbox.
- **Training Videos: Video training** on our products and tools, including a dedicated series on **PSoC 6 MCU** is available.

2.2 Firmware/Application Development

There are two development platforms that you can use for application development with PSoC 6 MCU:

- **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 ModusToolbox IDE.

  ModusToolbox IDE is an Eclipse-based development environment that runs on Windows, macOS, and Linux platforms and includes various tools.

  ModusToolbox supports stand-alone device and middleware configurators that are fully integrated into the 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 devices. It is recommended that you use ModusToolbox for all application development for PSoC 6 MCUs. See the **ModusToolbox Software Overview** for more information.

  Libraries and enablement software are available at the **GitHub** site. Some resources will be used by all developers. Others will be used by developers in particular ecosystems.

Development Ecosystem

Software resources available at GitHub support one or more of the target ecosystems:

- MCU and Bluetooth SoC ecosystem – a full-featured platform for PSoC 6, Wi-Fi, Bluetooth, and Bluetooth Low Energy application development
- Mbed OS ecosystem – provides an embedded operating system, transport security and cloud services to create connected embedded solutions
- Amazon FreeRTOS ecosystem – extends the FreeRTOS kernel with software libraries that make it easy to securely connect small, low-power devices to AWS cloud services

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

See **AN228571 – Getting Started with PSoC 6 MCU on ModusToolbox** for more information.

- **PSoC Creator:** A proprietary IDE that runs on Windows only. It supports a subset of PSoC 6 MCU devices as well as other PSoC device families such as PSoC 3, PSoC 4, and PSoC 5LP.

2.2.1 Choosing an IDE

ModusToolbox, the latest-generation toolset, includes the ModusToolbox IDE. The IDE is Eclipse-based and therefore is supported across Windows, Linux, and MacOS platforms. The tool supports all PSoC 6 MCU devices. The associated hardware and middleware configurators also work on all three host operating systems.

Certain features of PSoC 6 MCU such as UDBs and USB host are not currently supported in ModusToolbox IDE. New versions of ModusToolbox will be released in the future to support these features and improve the user experience.

Choose ModusToolbox if you have prior experience with Eclipse-based tools and want to take advantage of the power and extensibility of an Eclipse-based IDE, or if you want your development environment on Linux or macOS. You should also choose ModusToolbox if you want to build an IoT application using IoT devices, or if you are using a PSoC 6 MCU device not supported on PSoC Creator.

PSoC Creator is the long-standing proprietary tool that runs on Windows only. This mature IDE includes a graphical editor that supports schematic based design entry with the help of Components. PSoC Creator supports all PSoC 3, PSoC 4, PSoC 5LP devices and a subset of PSoC 6 MCU devices. The subset of PSoC 6 MCU devices includes devices up to 1 MB of flash.

Choose PSoC Creator if you are inclined towards using a graphical editor for design entry and code generation, and if the PSoC MCU that you are planning to use is supported by the IDE or if you are intending to use the UDBs on the PSoC MCU.

2.2.2 PSoC Creator

PSoC Creator is a free Windows-based Integrated Design Environment (IDE). It brings together several digital, analog, and system Components and firmware to build an application, and enables you to design hardware and firmware systems concurrently. Using PSoC Creator, you can select, place, and configure Components on a schematic; write C/assembly source code; and program and debug the device.

As **Figure 2** shows, with PSoC Creator, you can:

1. Browse the collection of code examples from the **File** > **Code Example…** menu.
   a) Filter for examples based on device family.
   b) Select from the menu of examples offered based on the **Filter by** options.
   c) Download the code example using the download button.
   d) Create a new project based on the selection.

2. Explore the library of more than 100 Components.
3. Drag and drop Components to build your hardware system design in the main design workspace.
4. Review the Component datasheets.
5. Configure the Components using configuration tools.
6. Co-design your application firmware with the PSoC hardware.

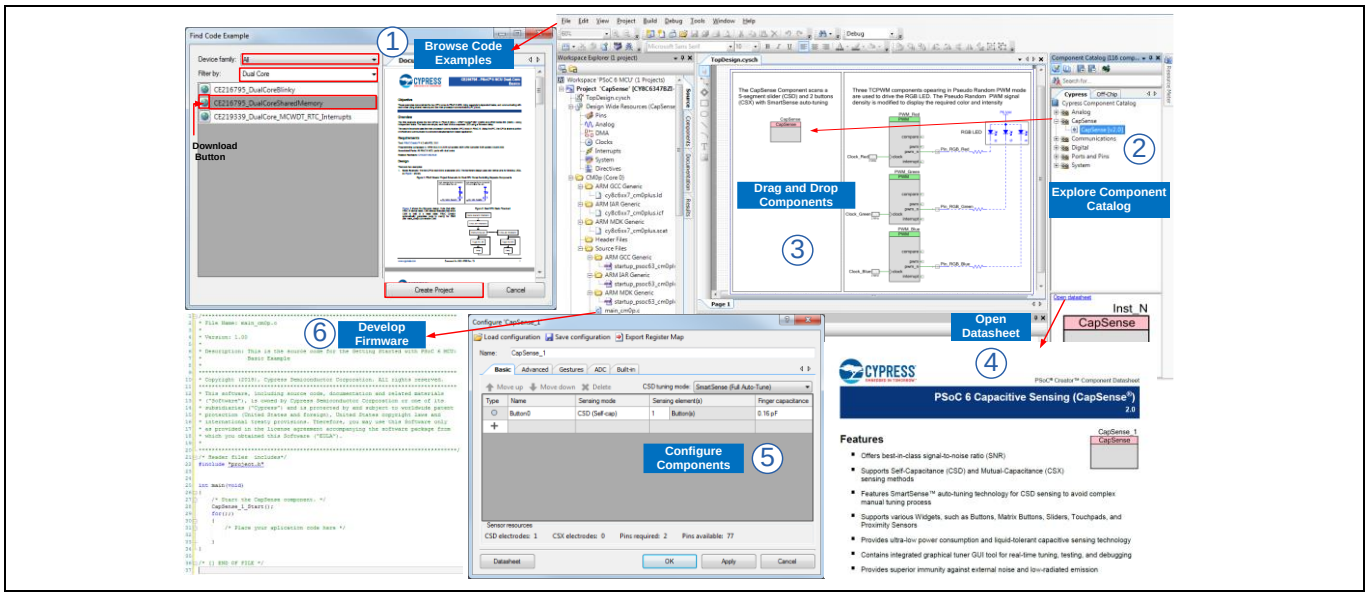

Figure 2 PSoC Creator Schematic Entry and Components

2.2.2.1 PSoC Creator Help

Visit the **PSoC Creator** home page to download and install the latest version of PSoC Creator. 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.
- **Code Examples**: Choose **File** > **Code Example** or click the **Find Code Example...** link on the **Start Page** tab. These code examples demonstrate how to configure and use PSoC resources.
- **Component Datasheets**: Right-click a Component and select **Open Datasheet**. Visit the **PSoC 6 MCU Component Datasheets** page for a list of all Component datasheets.

2.2.3 Software Development Kits for PSoC 6 Devices

Significant source code and tools are provided to enable software development for PSoC 6 MCU. 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 Bluetooth Low Energy (LE) connectivity or FreeRTOS. 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.

For the PSoC Creator environment, Infineon provides the Peripheral Driver Library (PDL). The PDL supports both PSoC Creator and third-party IDEs. You use PSoC Creator Components to configure the hardware. PSoC Creator generates configuration code based on your choices. That code is based on the source code in the PDL drivers. The PDL also includes various middleware libraries. There may or may not be a Component to assist in configuring that code.

Development Ecosystem

The driver code is delivered as the *psoc6pdl* library. Middleware is delivered as *psoc6mw*. The PDL source code is essentially identical, whether delivered with PSoC Creator or ModusToolbox IDE. There are implementation differences for the two IDEs.

There are differences in how the middleware is provided. For example, CapSense functionality is provided in PSoC Creator as a Component. For ModusToolbox software, there is a Configurator and a middleware library. See the respective documentation for the two IDEs for details of what's the same, and what's different.

Whether you use PSoC Creator or a third-party IDE, firmware developers who wish to work at the register level should also use the driver source code from the PDL. The PDL includes all the device-specific header files and startup code you need for your project. It also serves as a reference for each driver. Because the PDL is provided as source code, you can see how it accesses the hardware at the register level.

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

- All API elements needed to initialize, configure, and use a peripheral are available.
- The PDL is useful across various PSoC 6 MCU devices, regardless of available peripherals.
- The PDL includes error checking to ensure that the targeted peripheral is present on the selected device.

This enables the code to maintain compatibility across some members of the PSoC 6 device family as long as the peripherals are available. A device header file specifies the peripherals that are available for a device. If you write code that attempts to use an unsupported peripheral, you will get an error at compile time. Before writing code to use a peripheral, consult the datasheet for the particular device to confirm support for the peripheral.

PSoC Creator provides Components that are based on the PDL. This retains the essence of PSoC Creator in utilizing Infineon or community-developed and pre-validated Components. However, the PDL is a source code library that you can use with any development environment.

The PDL includes the following key software resources:

- Header and source files for each peripheral driver.
- Header and source files for middleware libraries.
- Device-specific header, startup, and configuration files.
- Template projects for supported third-party IDEs.
- Full documentation, available in *<PDL install directory>\doc*.

There are two key documents:

The PDL v3.x User Guide covers the fundamentals of working with the PDL, such as the following:

- Creating a custom project using the PDL (including third-party IDEs).
- Configuring a peripheral.
- Managing pins in firmware.
- Using the PDL as a learning tool for register-based programming.
- Using the PDL API Reference documentation.

The *PDL 3.x API Reference Manual.html*. This reference has complete information on every driver in the PDL, including overview, configuration considerations, and details on every function, macro, data structure, and enumerated type.

2.3 Support for Other IDEs

You can also develop firmware for PSoC 6 MCU using your favorite IDE such as **IAR Embedded Workbench**. It is recommended that you generate resource configuration using a configuration tool. For PSoC Creator, configuration is integral to the IDE.

PSoC Creator is used to set up and configure PSoC 6 MCU system resources and peripherals. You then export the project to your IDE, and continue developing firmware in your IDE. If there is a change in the device configuration, you edit the TopDesign schematic in PSoC Creator and regenerate the code for the target IDE.

You can work effectively in most if not all IDEs. If your IDE is not supported in the Target IDEs panel, you can still use PSoC Creator. After you generate code, add the necessary files directly to your IDE's project. **AN219434 – PSoC 6 MCU Importing Generated Code into an IDE** provides detailed steps for manually importing the generated code into another IDE.

2.4 RTOS Support

2.4.1 RTOS Support with PSoC Creator

The PDL includes RTOS support for PSoC 6 MCU development: FreeRTOS source code is fully integrated and included with the PDL. You can import the FreeRTOS software package into your project by using the PSoC Creator RTOS import option. Navigate to the **Project** > **Build Settings** menu and select **FreeRTOS** from the **Software package imports** option under **Peripheral Driver Library** > **FreeRTOS** as shown in **Figure 3**.

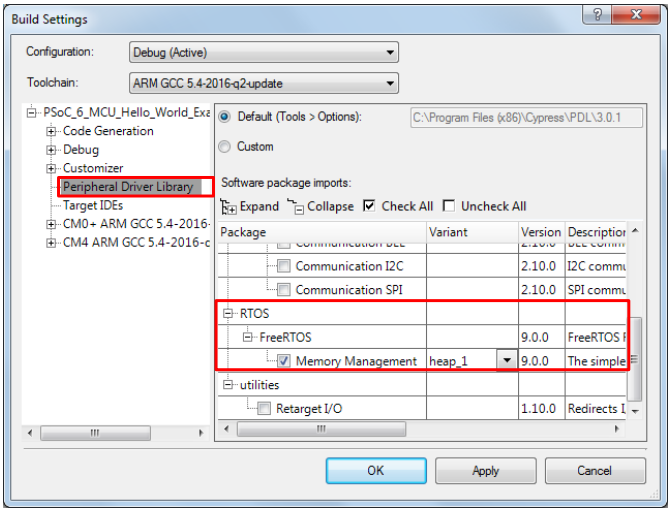

Figure 3 Import FreeRTOS in PSoC Creator Project

If you have a preferred RTOS, use the resources provided as examples on how to integrate such code with the PDL.

2.5 Programming/Debugging

The **PSoC 6 Wi-Fi-BT Pioneer Kit (CY8CKIT-062-WiFi-BT)** and **PSoC 6 BLE Pioneer Kit (CY8CKIT-062-BLE)** have the KitProg2 onboard programmer/debugger. It supports Cortex Microcontroller Software Interface Standard - Debug Access Port (CMSIS-DAP) and custom modes of operations, as well as the KitProg2 connection. This makes debugging the PSoC 6 MCU Pioneer Kit extremely flexible. See the **KitProg2 User Guide** for details.

Development Ecosystem

The **PSoC 6 Wi-Fi BT Prototyping Kit (CY8CPROTO-062-4343W)** has the KitProg3 onboard programmer/debugger. It supports Cortex Microcontroller Software Interface Standard - Debug Access Port (CMSIS-DAP). See the **KitProg3 User Guide** for details.

PSoC Creator supports debugging a single CPU (either Cortex-M4 or Cortex-M0+) at a time. Some third-party IDEs support multi-CPU debugging. For more information on debugging firmware on PSoC devices with PSoC Creator, refer to the PSoC Creator Help.

2.6 PSoC 6 MCU Development Kits

PSoC 6 Wi-Fi-BT Pioneer Kit (CY8CKIT-062-WiFi-BT) and **PSoC 6 Wi-Fi BT Prototyping Kit (CY8CPROTO-062-4343W)** are development kits that support the PSoC 62 series MCU along with Wi-Fi and Bluetooth connectivity.

The **PSoC 6 BLE Pioneer Kit (CY8CKIT-062-BLE)** and **PSoC 6 BLE Prototyping Kit (CY8CPROTO-063-BLE)** support the PSoC 6 MCU with Bluetooth LE Connectivity. Refer to the **PSoC 6 MCU product page** for more information.

3 Device Features

The PSoC 6 MCU device has an extensive feature set as shown in **Figure 4**. The following is a list of its major features. For more information, see the device **datasheet**, the **Technical Reference Manual (TRM)**, and the section on **Related Application Notes and Code Examples**.

- MCU subsystem
  - 150-MHz Arm Cortex-M4 and 100-MHz Arm Cortex-M0+
  - Up to 2 MB of flash with additional 32 KB for EEPROM emulation and 32-KB supervisory flash
  - Up to 1 MB of SRAM with selectable Deep Sleep retention granularity at 32-KB retention boundaries
  - Inter-processor communication supported in hardware
  - DMA controllers
- Security Features
  - Cryptography accelerators and true random number generator function
  - One-time programmable eFUSE for secure key storage
  - Secure boot with hardware hash-based authentication
- I/O subsystem
  - Up to 104 GPIOs with programmable drive modes, drive strength, slew rates
  - Two ports with Smart I/O that can implement Boolean operations
- Programmable analog blocks
  - Two opamps of 6-MHz gain bandwidth (GBW) and two low-power comparators
  - Up to One 12-bit, 1-Msps SAR ADC and one 12-bit voltage-mode DAC
- Programmable digital blocks, communication interfaces
  - Up to 12 UDBs for custom digital peripherals
  - Up to 32 TCPWM blocks configurable as 16-bit/ 32-bit timer, counter, PWM, or quadrature decoder
  - Up to 13 SCBs configurable as I2C Master or Slave, SPI Master or Slave, or UART
  - Controller Area Network interface with Flexible Data-Rate
  - Up to two Secure Digital Host Controllers with support for SD, SDIO, and eMMC interfaces
  - Audio subsystem with up to two I2S interface and two PDM channels
  - SMIF interface with support for execute-in-place from external quad SPI flash memory and on-the-fly encryption and decryption
  - Full-Speed, dual-role USB with device and host capability
- CapSense with SmartSense™ auto-tuning
  - Supports both CapSense Sigma-Delta (CSD) and CapSense Transmit/Receive (CSX) controllers
  - Provides best-in-class SNR, liquid tolerance, and proximity sensing
- Operating voltage range, power domains, and low-power modes
  - Device operating voltage: 1.71 V to 3.6 V with user-selectable core logic operation at either 1.1 V or 0.9 V
  - Multiple on-chip regulators: low-drop out (LDO for Active, Deep Sleep modes), buck converter
  - Six power modes for fine-grained power management
  - An “Always ON” backup power domain with built-in RTC, power management integrated circuit (PMIC) control, and limited SRAM backup

Device Features

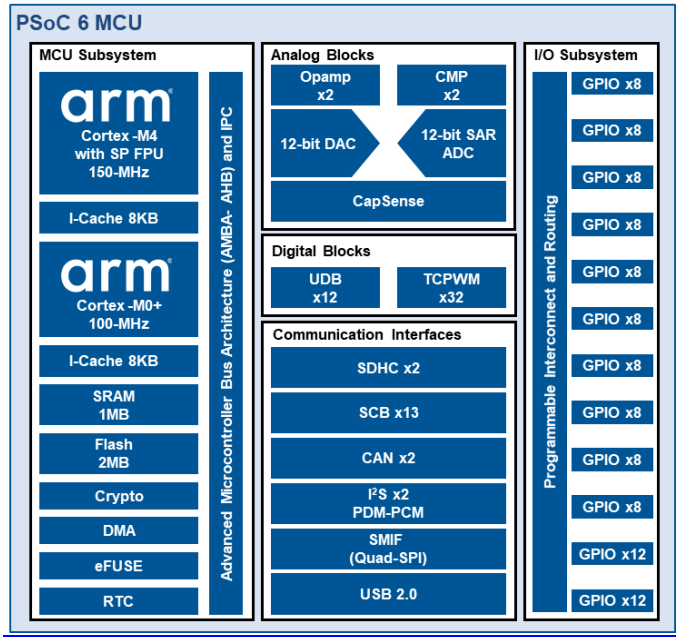

Figure 4 PSoC 6 MCU Block Diagram

4 My First PSoC 6 MCU Design Using PSoC Creator

This section does the following:

- Demonstrates how to build a simple PSoC 6 MCU-based design and program it on to the development kit.
- Provides detailed steps that make it easy to learn PSoC 6 MCU design techniques and how to use the PSoC Creator IDE.

4.1 Using These Instructions

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

- **Part 1: Create a New Project from Scratch**
- **Part 2: Implement the Design**
- **Part 3: Generate Source Code**
- **Part 4: Write the Firmware**
- **Part 5: Build the Project and Program the Device**
- **Part 6: Test Your Design**

If you are familiar with developing projects with PSoC Creator, you can use the PSoC Creator version of the code example **CE221773 – PSoC 6 MCU Hello World Example** directly. It is a complete design, with all the firmware written. You can walk through the instructions and observe how the steps are implemented in the code example.

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

You can download the code example from the website by clicking the link above. You can also use the PSoC Creator **File** > **Code Example** command. Set the **Device family** to PSoC 62. Select the PSoC MCU Hello World Example. Download the code example by clicking on the download icon adjacent to the example and then click on **Create Project**, and follow the on-screen instructions.

This design is developed for the **CY8CKIT-062-WiFi-BT PSoC 6 Wi-Fi-BT Pioneer Kit**. You can also use **CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit** to test this example by selecting the appropriate device from the Device Selector.

4.2 About the Design

This design uses the CM4 CPU of PSoC 6 MCU to execute two tasks: UART communication and LED control. At device reset, the CM0+ CPU enables the CM4 CPU. The CM4 CPU uses the UART Component to print a “Hello World” message to the serial port stream and when the Enter Key is pressed by the user, the LED on the PSoC 6 MCU Wi-Fi-Bluetooth Pioneer Kit starts blinking.

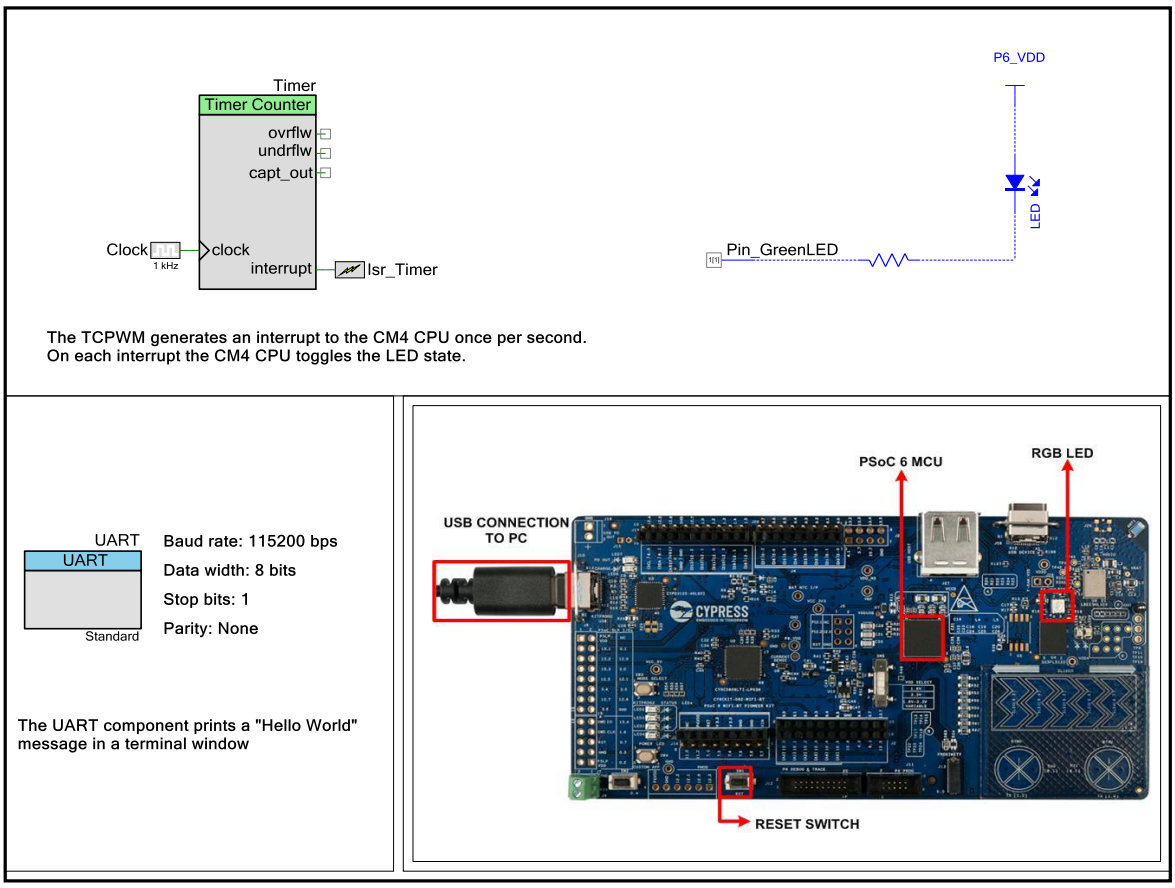

Figure 5 My First PSoC 6 MCU Design

4.3 Part 1: Create a New Project from Scratch

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 development stages.

Note: These instructions assume that you are using PSoC Creator 4.2. The overall development process is the same for subsequent versions of PSoC Creator; but the user interface may change over time.

Launch PSoC Creator and get started.

1. **Ensure that PSoC Creator can find the PDL.**
   This should be set correctly automatically during installation, but nothing works if this isn't set up right. Refer to **Figure 6** for help with this step.
   a) Choose **Tools** > **Options**.
   b) On the **Project Management** panel, check the path in the **PDL v3 (PSoC 6 Devices) location** field.
   c) Ensure that it is correct. If it is not, click the **Browse** button and locate the installed directory of the PDL. The default location is *C:\Program Files (x86)\Cypress\PDL\3.0.1*.

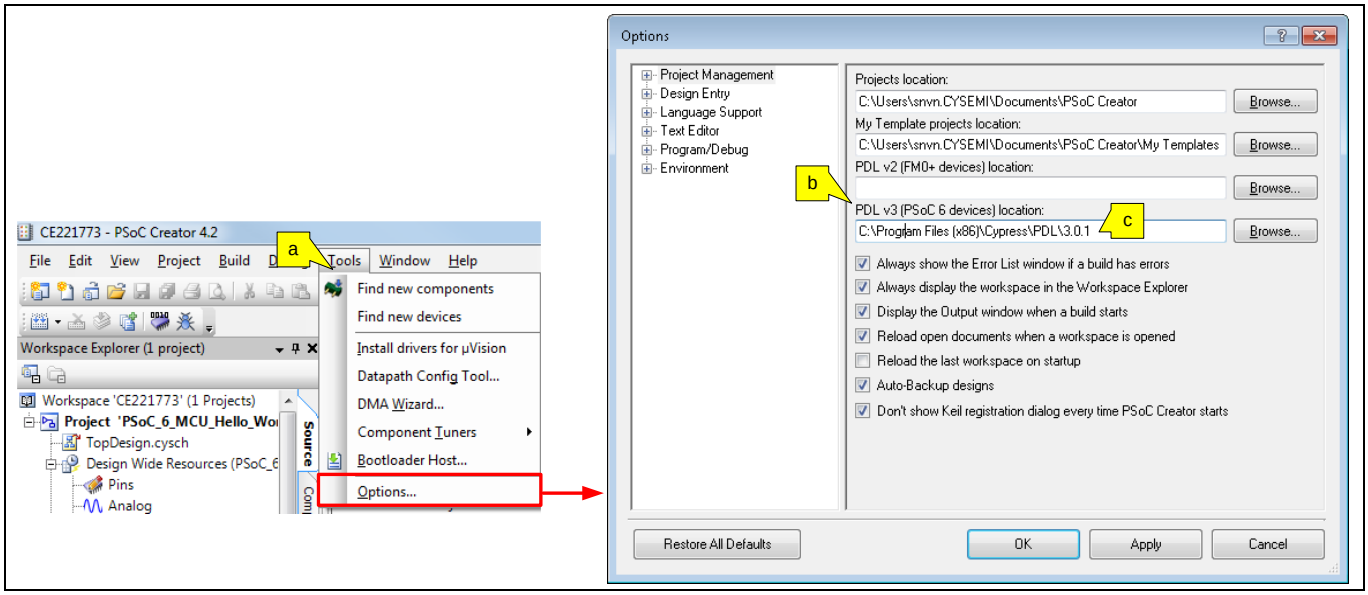

Figure 6 Peripheral Driver Library (PDL) Location

Optional: Jump to **Part 2: Implement the Design**.

2. **Create a new PSoC Creator project.**

   Choose **File** > **New** > **Project**, as **Figure 7** shows. The **Create Project** window appears.

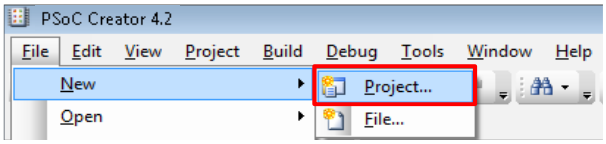

Figure 7 Create a New PSoC Creator Project

*Note: If you are using the code example, choose **File** > **Open** > **Project/Workspace**, as **Figure 8** shows. The **Open** window appears. Point to the location of the code example workspace and open the workspace.*

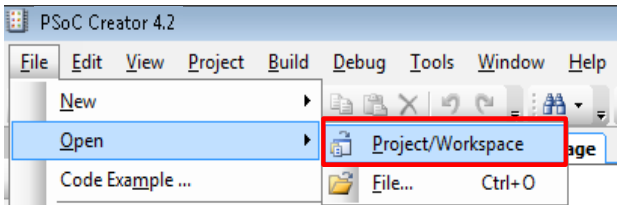

Figure 8 Open Existing Code Example Workspace

3. **Select PSoC 6 MCU as the target device.**

   PSoC Creator speeds up the development process by automatically setting various project options for specified development kits or target devices. See **Figure 9** for help with this step.

   a) Click **Target device**.

   b) In the family drop-down menu, select **PSoC 6**.

   c) In the device drop-down menu, select **PSoC 62**.

d) Click **Next**. The Select project template panel appears.

PSoC Creator uses CY8C6247BZI-D54 as the default device in the PSoC 6 MCU family. This device is mounted on the **CY8CKIT-062-WiFi-BT PSoC 6 Wi-Fi-BT Pioneer Kit**.

If you are using custom hardware based on PSoC 6 MCU, or a different PSoC 6 MCU part number, this is the place you choose to **Launch Device Selector** option in **Target device** and select the appropriate part number.

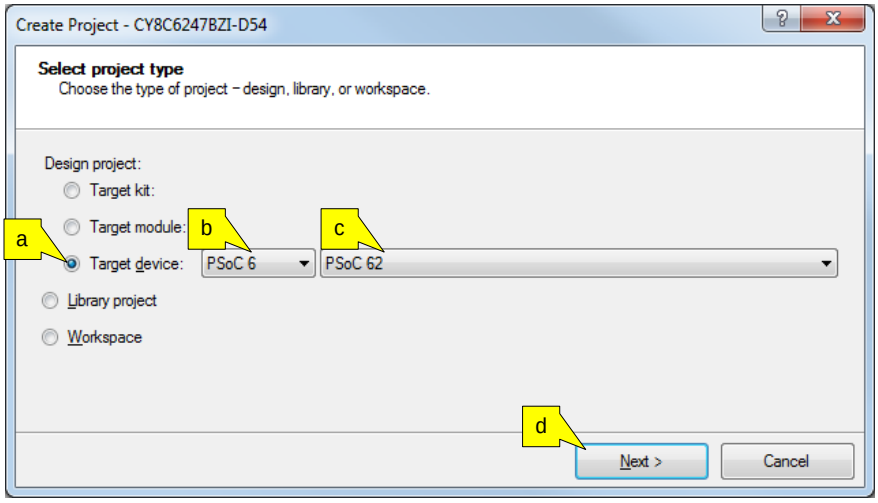

Figure 9 Selecting Target Device

4. Pick a project template.

a) Choose **Empty Schematic**.

b) Click **Next**.

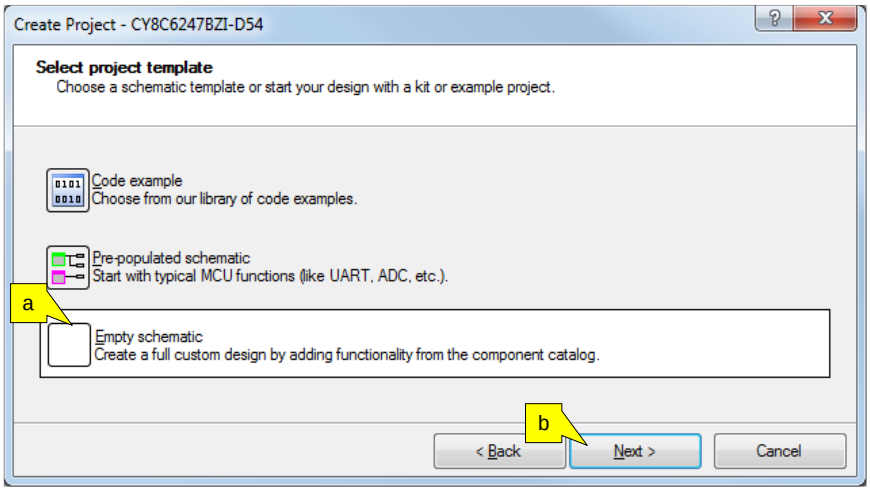

Figure 10 Pick a Project Template

5. Select target IDE(s).

If you expect to export the code from the project, specify the target IDE. By default, all export options are disabled. You can modify this setting later if circumstances change.

Click **Next** to accept the default options.

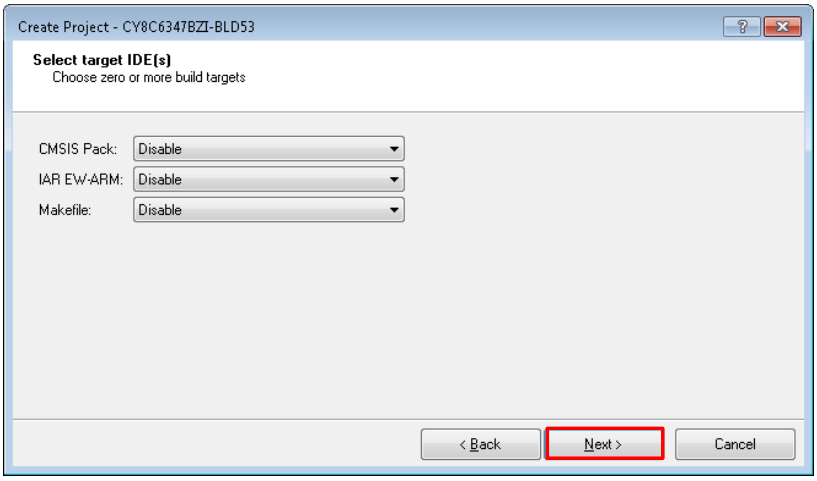

Figure 11 Select Target IDEs (All Disabled)

6. **Create the project.**

   In this step, you set the name and location for your workplace, and a name for the project. See **Figure 12** for help with this step. A workspace is a container for one or more projects.

   a) Set the Workspace name.

   b) Specify the **Location** of your workspace.

   c) Set a **Project name**. The project and workspace names can be the same or different.

   d) Click **Finish**.

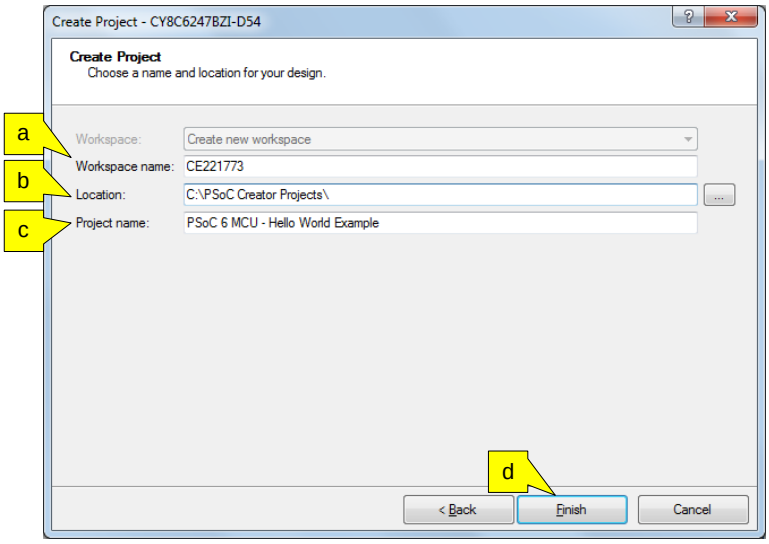

Figure 12 Project Naming and Location

You have successfully created a new PSoC Creator project.

4.4 Part 2: Implement the Design

Now that you have a project file, it is time to implement the hardware design using PSoC Creator Components. If you are using the code example directly, you already have a complete design.

Before you implement the design, a quick tour of the PSoC Creator interface is in order.

My First PSoC 6 MCU Design Using PSoC Creator

Figure 13 shows the PSoC Creator application displaying an empty design schematic.

The project includes a project folder with a base set of files. You view these files in the **Workspace Explorer** pane to the left. The project schematic opens by default. This is the *TopDesign.cysch* file. Double-click the file name in the explorer pane to open the schematic at any time. In a new project, the schematic is empty. If you are using the code example, this is the schematic for the design.

The Component catalog is on the right side of the window. You can open it with the **View** > **Component Catalog** menu item. You can search for a particular Component by typing the name of the Component in the **Search for…** text box and then pressing the enter key. See **Figure 13**.

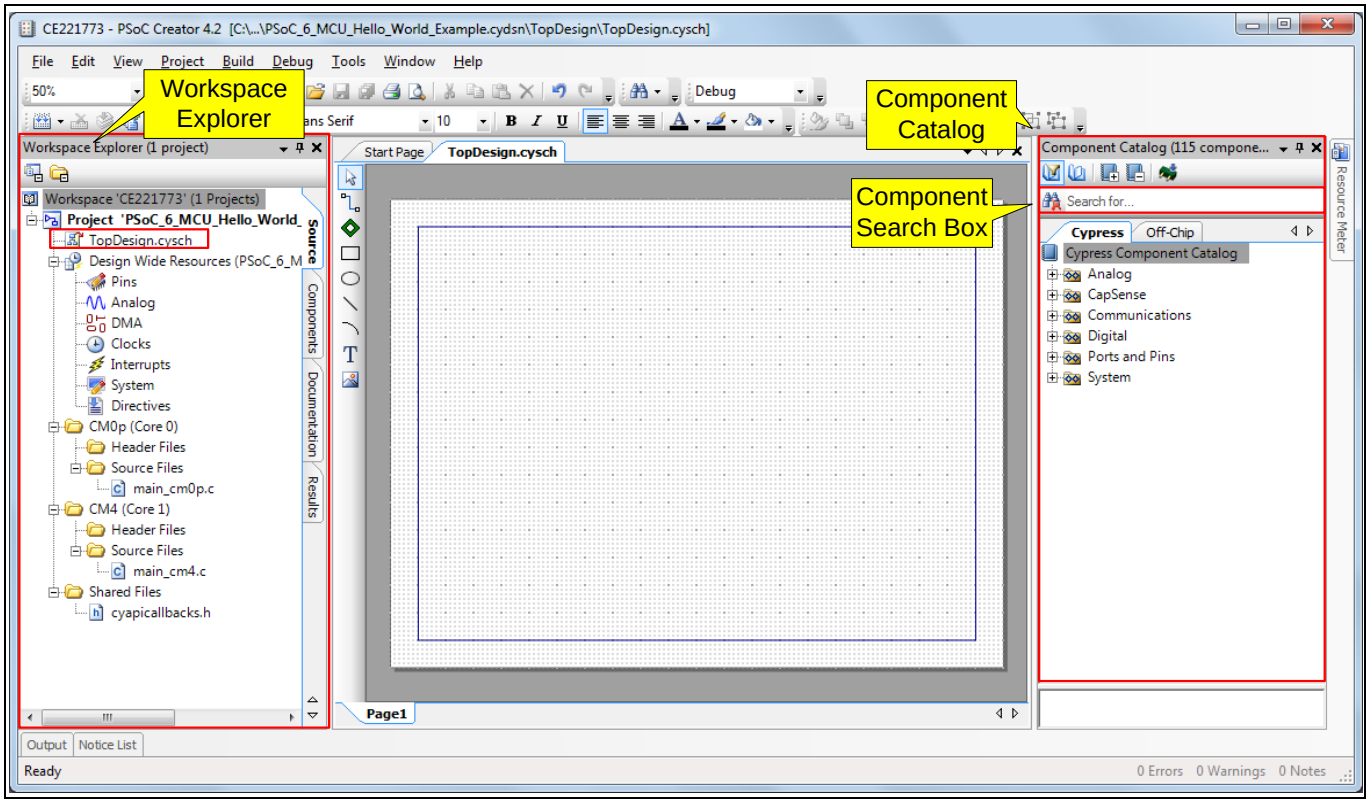

Figure 13 Schematic and Component Catalog

1. **Place Components in the design.**

   This design uses several Components: three digital output pins, a UART, a Watchdog Timer, and an Interrupt. In this step, you add them to the design. You configure them in subsequent steps. **Figure 14** shows the result.

   a) In the **Component Catalog**, expand the **Communications** group, drag a **UART (SCB)** Component into the schematic, and drop it. It doesn't matter where you put a Component.

   b) Expand the **Ports and Pins** group, and drag a **Digital Output Pin** into the design.

   c) Expand the **Digital** group, and drag a **Timer Counter** (TCPWM) Component into the design.

   d) Expand the **System** group, and drag an **Interrupt** Component and a **Clock** Component into the design.

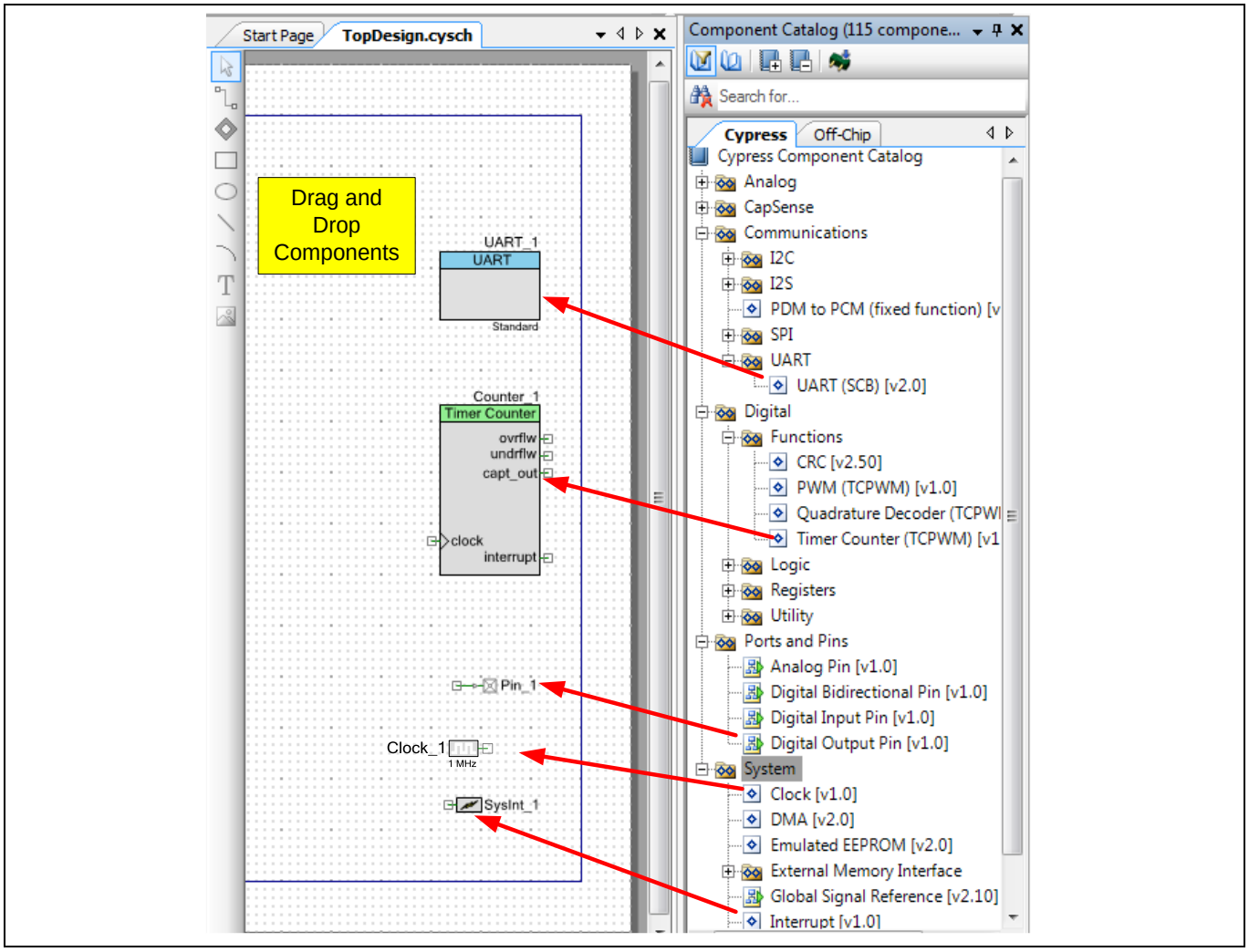

Figure 14 Place Components in the Design

PSoC Creator gives each Component a default name and properties. Default values may or may not be suitable for any given design. In subsequent steps, you modify the name and some of the properties.

2. **Configure the LED pin.**

   The output pin drives the LED. The LED on the PSoC 6 Wi-Fi-BT Pioneer Kit is active LOW; that is, the logic HIGH pin-drive state turns OFF the LED, and the logic LOW pin-drive state turns it ON. **Figure 15** shows the configuration.

   Double-click the Component placed on the schematic to open the configuration dialog. Then, perform the following steps:

   a) Change the name of the Component instance to **Pin_GreenLED**.

   b) Deselect **HW connection**. The firmware will drive the pin.

   c) Set the **Drive Mode** to Resistive Pull Up.

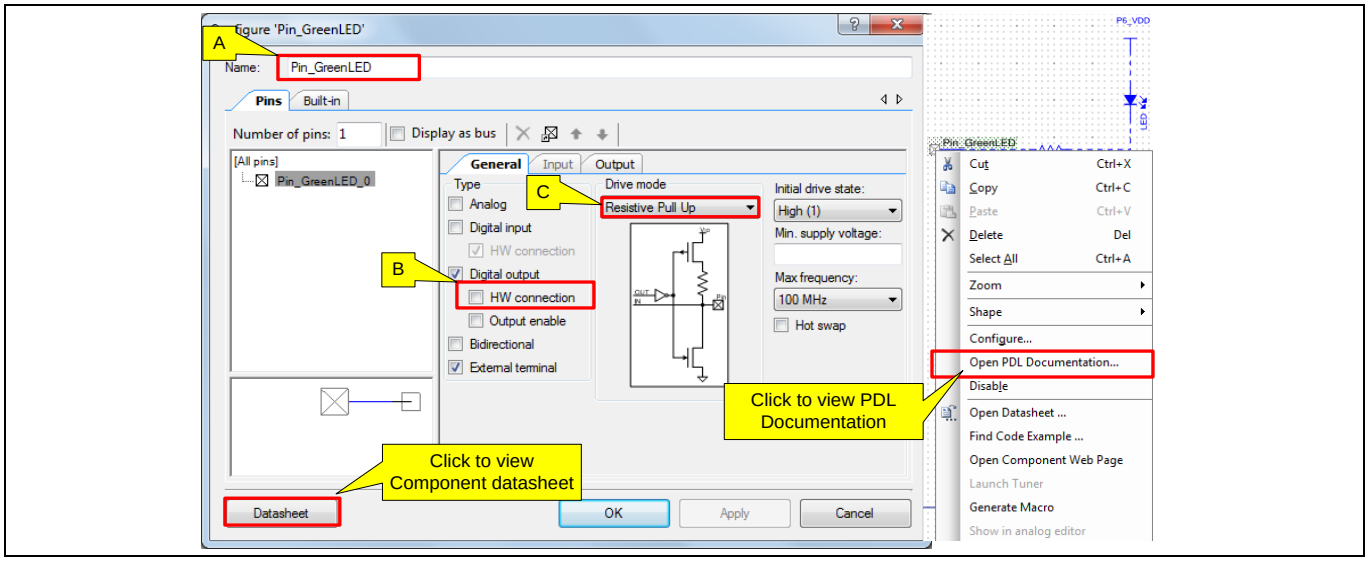

Figure 15 Configuring an Output Pin Component

Note:

1. *Each Component has an associated datasheet that can be accessed from the configuration window. The Component datasheet provides more information on the Component configuration, the application programming interface (API), and the electrical specifications.*
2. *You can open the API reference document of the associated PDL driver of a Component by right-clicking the Component and clicking on **Open PDL Documentation…** link. See **Figure 15**.*
3. *For a pin, if you enable **External terminal**, you can add external "off-chip" Components to a design. External Components on the schematic are included for descriptive purposes only; they have no effect on the generated code. Off-chip Components are optional, but can assist the hardware design team understanding how the design works. You can also add text boxes to a design with descriptions. **Figure 16** shows how you could enhance the design for the LED. In this case, the off-chip components were configured with the **Instance_Name_Visible** option unchecked. The resistor was configured with the **Value** field left blank. The power terminal was configured with the **Supply_Name** set to P6_VDD.*

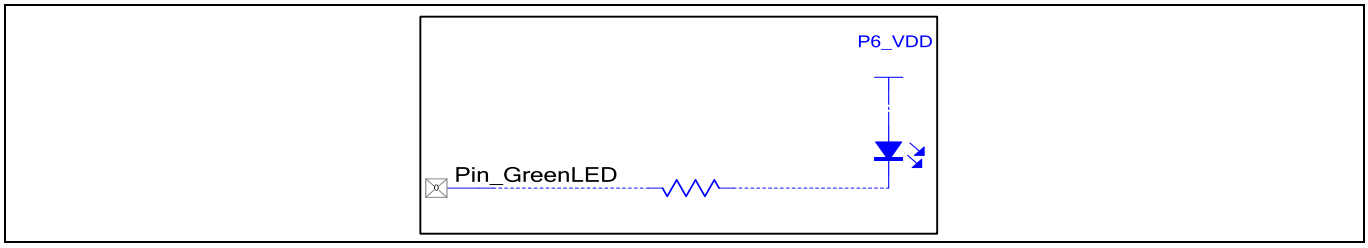

Figure 16 An Output Pin with Off-Chip Components

3. Configure the UART Component.

Double-click the Component to open the configuration window. The design uses this Component to display messages in a terminal window at a baud rate of 115200 bps.

a) Change the **Name** of the Component instance to **UART**.

b) Click **OK**.

The design uses default values for all other settings.

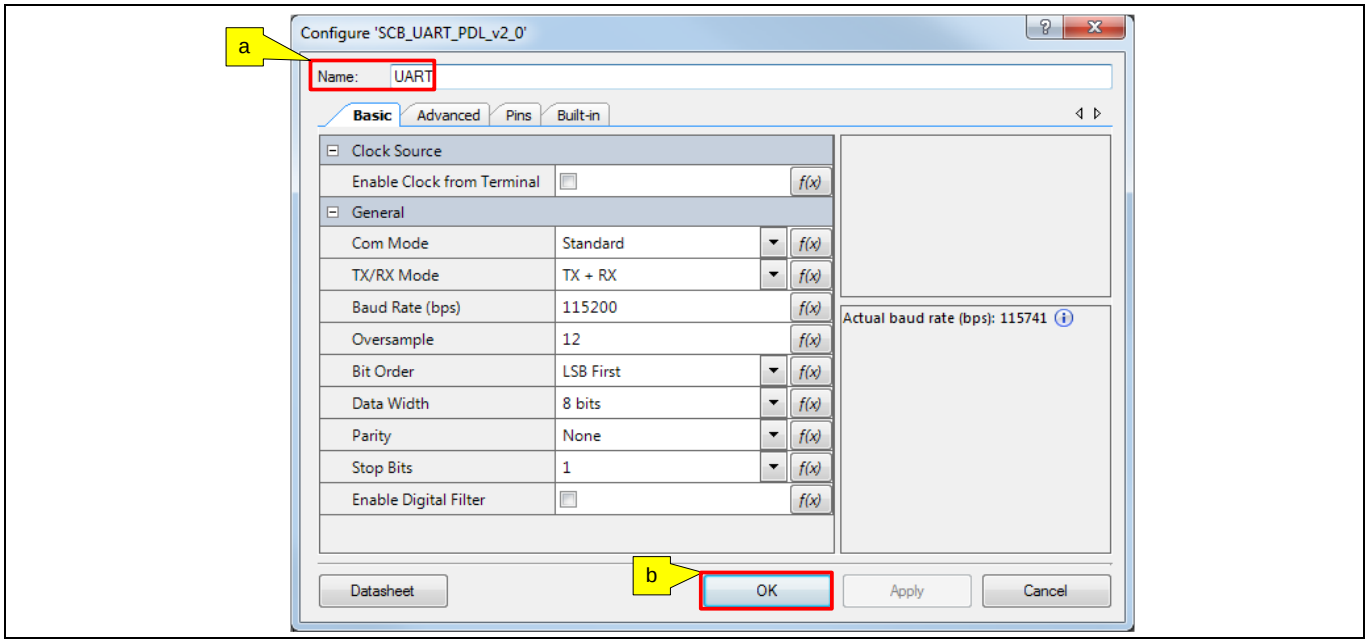

Figure 17 Configuring the SCB-Based UART Component

4. Configure the Timer Counter (TCPWM) Component to trigger an interrupt.

In this step, you configure the Timer Counter (TCPWM) Component to trigger an interrupt every second (1 Hz). The clock source of the TCPWM is the peripheral clock (Clk_Peri). The design will use this interrupt to toggle the LED state. Open the Component customizer and follow the steps illustrated in **Figure 18**.

a) Change the **Name** to **Timer**.

b) Set the **Period** to 1000 and **Interrupt Source** as Overflow/Underflow.

c) Click **OK** to complete the configuration of the TCPWM Component.

d) Connect the clock terminal of the TCPWM to the 1-kHz clock source. In the schematic, use the wire tool button or press the 'W' key to start wiring the Clock Component to the clock terminal of the TCPWM Component.

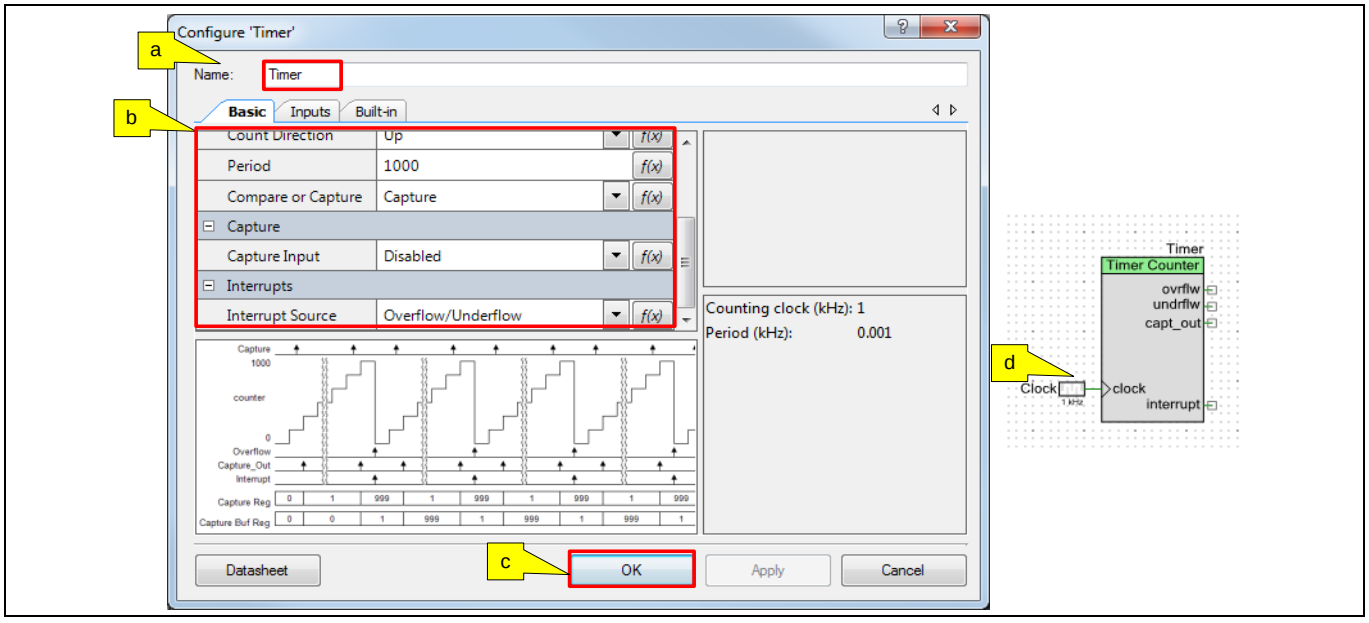

Figure 18 Configuring the TCPWM Component

5. Configure the interrupt Component.

In this step, you configure the SysInt Component to map the TCPWM interrupt to the CM4 CPU. Open the Component customizer and follow the steps illustrated in **Figure 19**.

a) Change the **Name** to **Isr_Timer**.

b) Click **OK** to complete the configuration of the SysInt Component.

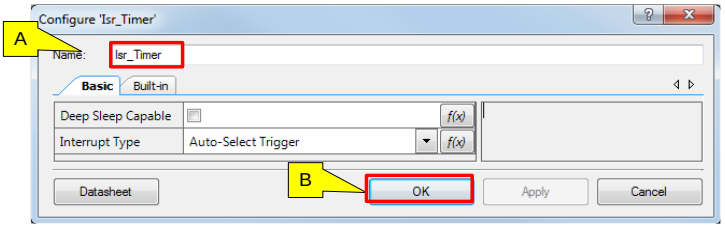

Figure 19 SysInt_PDL Settings

As the final step, connect the interrupt output of the TCPWM Component to the Isr_TCPWM Component input. This routes the TCPWM interrupt to the CM4 CPU (the selection of the CM4 CPU for this interrupt will be set in the system interrupt configuration in a later step). In the schematic, use the wire tool button or press the 'W' key to start wiring the Components.

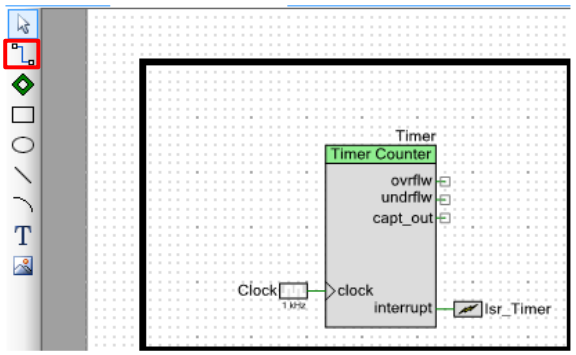

Figure 20 Connect TCPWM Peripheral Interrupt to CM4 CPU

6. Set the physical pins for each Pin Component.

One task remains to complete the design. You must associate each Component with the required physical pins on the device. The choice of which pin to use is driven by the board design. You can find this information in the kit schematic. **Figure 21** shows the result of this step. You can connect external LEDs to the selected pins.

To set a pin, type either the port number or pin number in the corresponding field, or use the drop-down menu to pick the port or pin. Typically, the port number is used instead of the pin number since these names are independent of the specific package being used.

a) Open the pin selector.

In the **Workspace Explorer** pane, double-click the **Pins** item under the Design Wide Resources. The pin selector for this device appears.

b) Set each pin as shown in **Table 1**.

Table 1 Physical Pin Assignments for CY8CKIT-062-WiFi-BT Pioneer Kit

Pin Component Name	Port Name
UART: rx	P5[0]
UART: tx	P5[1]
Pin_GreenLED	P1[1]

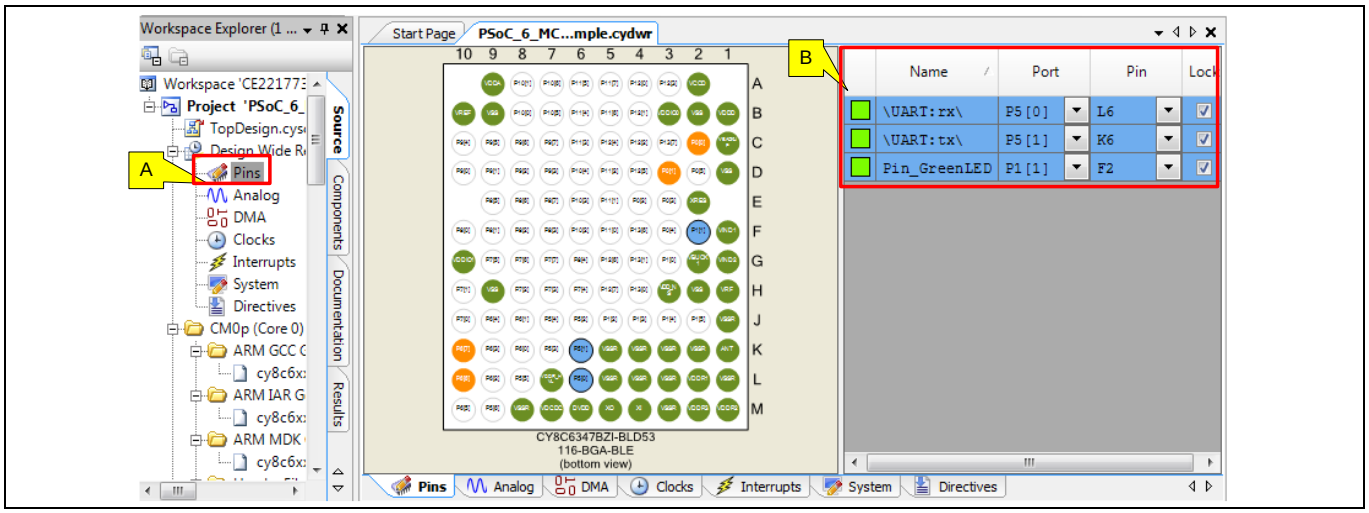

Figure 21 Pin Assignment

7. Configure System Clock.

The design uses default values for the high-frequency system clock settings. Although you do not modify high frequency clocks for this design, you should know how PSoC Creator manages them. If you are working with your own board, you may need to modify these clocks.

a) In the **Workspace Explorer** pane, double-click the **Clocks** item under **Design Wide Resources**. The list of clocks appears.

b) Click **Edit Clock**. The **Configure System Clocks** dialog appears.

Here, you can see the clock tree, and modify the clocks as required. Note that there are tabs for different types of clocks such as **Source Clocks**, **FLL/PLL**, **High Frequency Clocks**, and **Miscellaneous Clocks**.

c) Click on the **FLL/PLL** tab. By default, P**SoC Creator enables FLL** and sets the frequency to 100 MHz.

d) Click on the **High Frequency Clocks** tab.

e) You can set the CM4 CPU clock by setting the divider in **Clk_Fast**. By default, the divider is set to 1.

f) You can set the CM0+ CPU clock by setting the divider in **Clk_Slow**. By default, the divider is set to 1. See **Figure 22**.

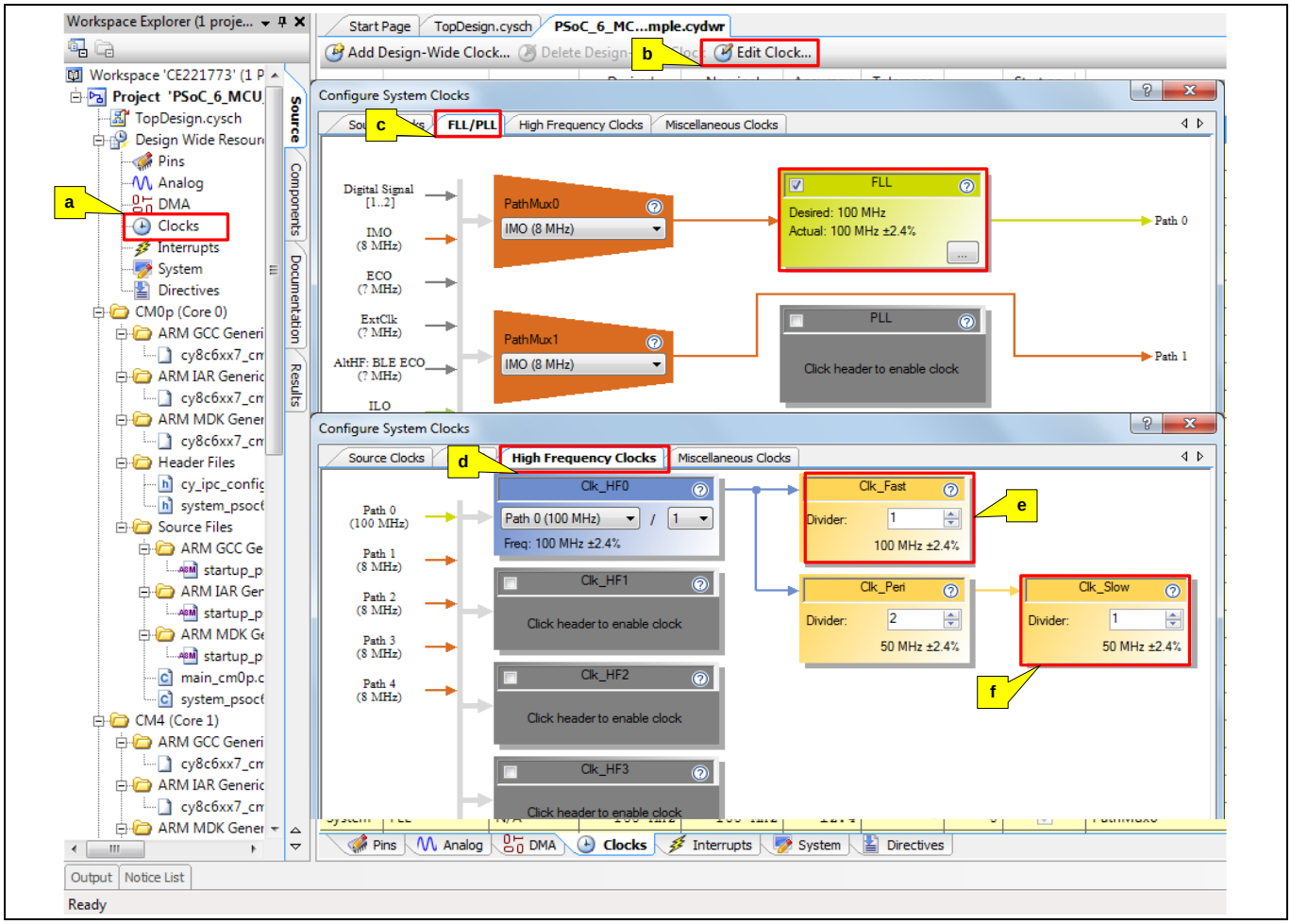

Figure 22 Clock Configuration

8. Configure System Interrupts.

In this step, you configure the system interrupts. See **Figure 23**.

a) In the **Workspace Explorer** pane, double-click the **Interrupts** item under **Design Wide Resources**. The list of interrupts appears.

b) Enable **Isr_Timer** for the CM4 CPU.

The interrupt numbers are generated automatically by PSoC Creator when you generate the code in **Part 3: Generate Source Code**.

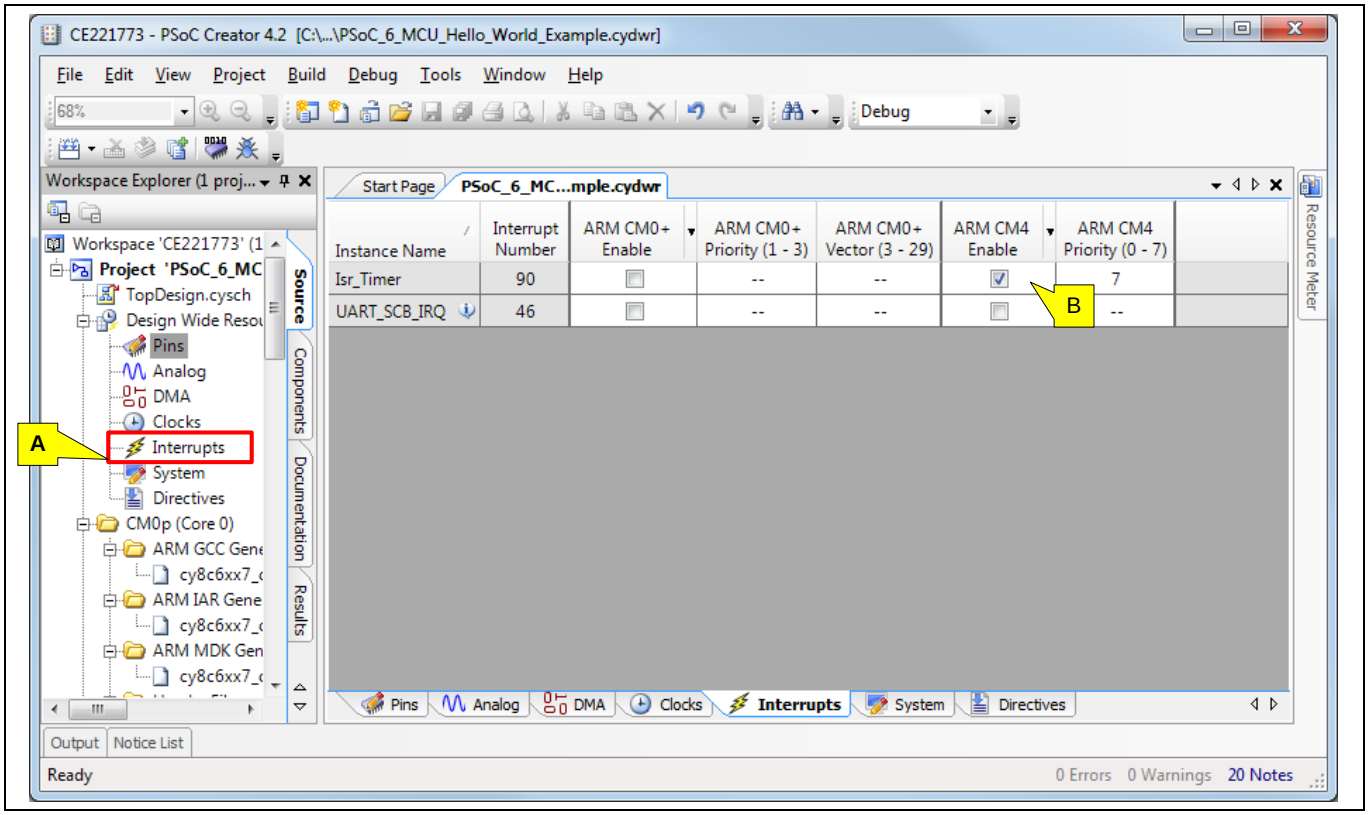

Figure 23 Interrupt Configuration

The next part in the development process is to generate code.

Note: This exercise does not detail how to export your work to a target IDE. However, if you wish to use a target IDE, this is the point in the workflow where you would ensure that the correct target IDE is selected before you generate the source code.

4.5 Part 3: Generate Source Code

PSoC Creator generates the source code based upon the design. The recommended workflow is to generate code before writing firmware. PSoC Creator will automatically create macros, constants, and API calls that you may then use in your firmware.

1. **Generate the application.**

   Choose **Build** > **Generate Application**. PSoC Creator generates the source code based on the design and puts the files in the *Generated_Source* folder. See **Figure 24**. PSoC Creator will alert you to errors or problems that may occur. If you are working from scratch and encounter errors, revisit the configuration steps in **Part 2: Implement the Design** to ensure you have performed them correctly.

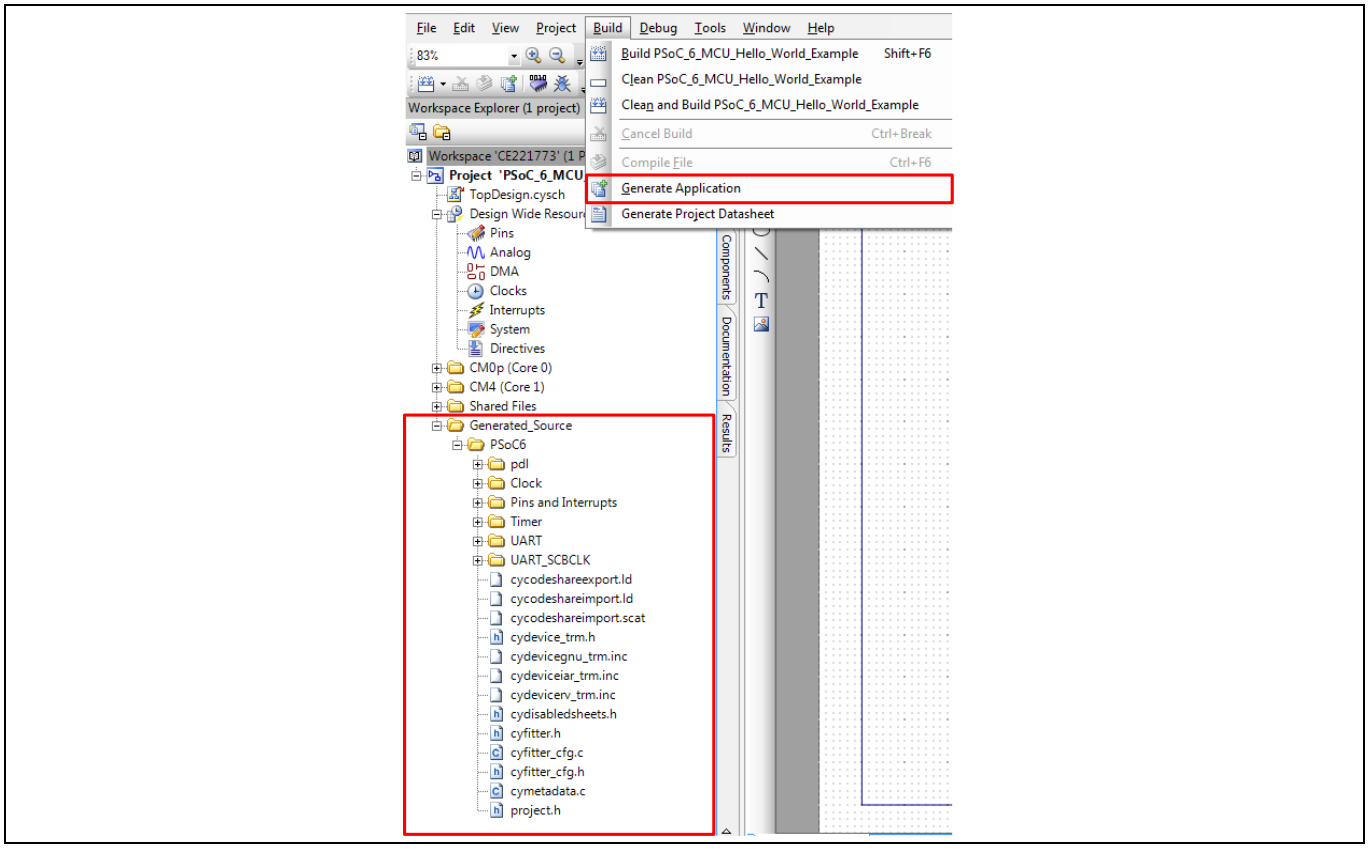

Figure 24 Generate Application

Background: PSoC 6 MCU is a dual-CPU platform. You can target firmware to run either on Cortex-M4 or Cortex-M0+. You set this at the source file level by accessing the file properties. Right-click on a source file, and select **Properties**. **Figure 25** shows the **Properties** dialog window. By default, the *main_cm0p.c* file is targeted to the Cortex-M0+ and the *main_cm4.c* file is targeted to the Cortex-M4. You do not need to modify the properties for any other file. They are already set in the code example.

By convention, files targeted to run on the CM0+ CPU are located in the *CM0p* folder and files targeted to run on the CM4 CPU are located in the *CM4* folder.

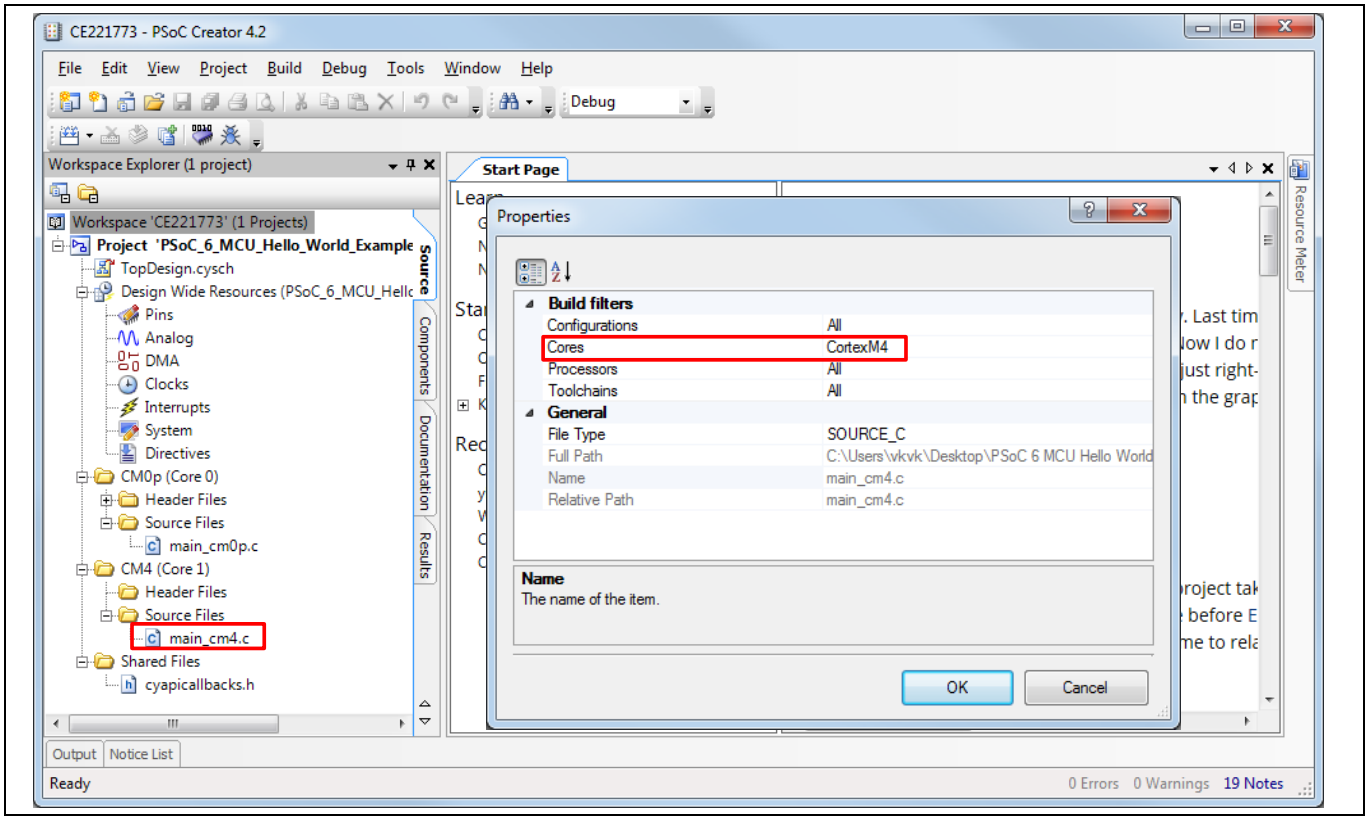

Figure 25 Setting Target Processor for a Source C File

4.6 Part 4: Write the Firmware

At this point in the development process, you have created a project, implemented a hardware design, and generated the code. In this part, you write the firmware that implements the design functionality.

The steps in this part discuss the firmware for the design that you configured in **Part 2: Implement the Design**.

The code example has all the required code. If you are working from scratch, you can copy the respective source codes to *main_cm0p.c* and *main_cm4.c* from the code snippet provided in this section. If you are using the code example, files are already in your project.

Firmware Flow

In the remaining steps, we examine code in the *main_cm0p.c* and *main_cm4.c* file.

When the PSoC 6 MCU device is reset, the firmware first performs system initialization, which includes setting up the CPUs for execution, enabling global interrupts, and enabling other Components used in the design.

The initialization is split across the CPUs. The CM0+ CPU comes out of reset and enables the CM4 CPU. The CM0+ CPU code snippet is given in **Code Listing 1**. Copy the following code snippet to the *main_cm0p.c* file of your project.

Code Listing 1

```
/* Header files includes*/
#include "project.h"
int main(void)
{
```

Code Listing 1

```
    __enable_irq(); /* Enable global interrupts. */

    /* Enable CM4.  CY_CORTEX_M4_APPL_ADDR must be updated
       if CM4 memory layout is changed. */
    Cy_SysEnableCM4(CY_CORTEX_M4_APPL_ADDR);

    for(;;)
    {

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

When the CM4 CPU is enabled, the UART Component is started and prints a “Hello World!” message on the terminal emulator. A Timer Counter PWM (TCPWM) Component is configured to generate an interrupt every second. At each interrupt, the CM4 CPU toggles the LED (**LED5**) state on the kit. Copy the code snippet in **Code Listing 2** to *main_cm4.c* of your project.

Code Listing 2

```
/* Header files includes*/
#include "project.h"
/*******************************************************************************
* Macros
*******************************************************************************/
#define LED_ON     (0)
#define LED_OFF    (!LED_ON)
/*******************************************************************************
* Function Prototypes
*******************************************************************************/
void UartInit(void);
void TimerInit(void);
void Isr_Timer(void);

/*******************************************************************************
* Global Variables
*******************************************************************************/
bool LEDupdateFlag = false;
/*******************************************************************************
* Function Name: main
*******************************************************************************/
int main(void)
{
    /* Start the UART peripheral */
    UartInit();

    /* Enable global interrupts. */
    __enable_irq();

    /* \x1b[2J\x1b[;H - ANSI ESC sequence for clear screen */
    Cy_SCB_UART_PutString(UART_HW, "\x1b[2J\x1b[;H");

    Cy_SCB_UART_PutString(UART_HW, "*****************CE221773 - PSoC 6 MCU:"\
```

Code Listing 2

```
                    " Hello World! Example*****************\r\n\n");

    Cy_SCB_UART_PutString(UART_HW, "Hello World!!!\r\n\n");

    Cy_SCB_UART_PutString(UART_HW, "Press Enter key to start blinking the LED\r\n\n");

    /* Wait for the user to Press Enter key */
    while(Cy_SCB_UART_Get(UART_HW) != '\r');

    /* Start the TCPWM peripheral. TCPWM is configured as a Timer */
    TimerInit();

    Cy_SCB_UART_PutString(UART_HW, "Observe the LED blinking on the kit!!!\r\n");

    for(;;)
    {
        if(LEDupdateFlag)
        {
            /* Clear the flag */
            LEDupdateFlag = false;

            /* Invert the LED state*/
            Cy_GPIO_Inv(Pin_GreenLED_0_PORT, Pin_GreenLED_0_NUM);
        }
    }
}
/*******************************************************************************
* Function Name: UartInit
*******************************************************************************/
void UartInit(void)
{
    /* Configure the UART peripheral.
       UART_config structure is defined by the UART_PDL component based on
       parameters entered in the Component configuration*/
    Cy_SCB_UART_Init(UART_HW, &UART_config, &UART_context);

    /* Enable the UART peripheral */
    Cy_SCB_UART_Enable(UART_HW);
}
/*******************************************************************************
* Function Name: TimerInit
*******************************************************************************/
void TimerInit(void)
{
    /* Configure the TCPWM peripheral.
       Counter_config structure is defined based on the parameters entered
       in the Component configuration */
    Cy_TCPWM_Counter_Init(Timer_HW, Timer_CNT_NUM, &Timer_config);

    /* Enable the initialized counter */
    Cy_TCPWM_Counter_Enable(Timer_HW, Timer_CNT_NUM);

    /* Start the enabled counter */
```

Code Listing 2

```
    Cy_TCPWM_TriggerStart(Timer_HW, Timer_CNT_MASK);

    /* Configure the ISR for the TCPWM peripheral*/
    Cy_SysInt_Init(&Isr_Timer_cfg, Isr_Timer);

    /* Enable interrupt in NVIC */
    NVIC_EnableIRQ((IRQn_Type)Isr_Timer_cfg.intrSrc);
}
/*******************************************************************************
* Function Name: Isr_Timer
*******************************************************************************/
void Isr_Timer(void)
{
    /* Clear the TCPWM peripheral interrupt */
    Cy_TCPWM_ClearInterrupt(Timer_HW, Timer_CNT_NUM, CY_TCPWM_INT_ON_TC );

    /* Clear the CM4 NVIC pending interrupt for TCPWM */
    NVIC_ClearPendingIRQ(Isr_Timer_cfg.intrSrc);
       LEDupdateFlag = true;
}
/* [] END OF FILE */
```

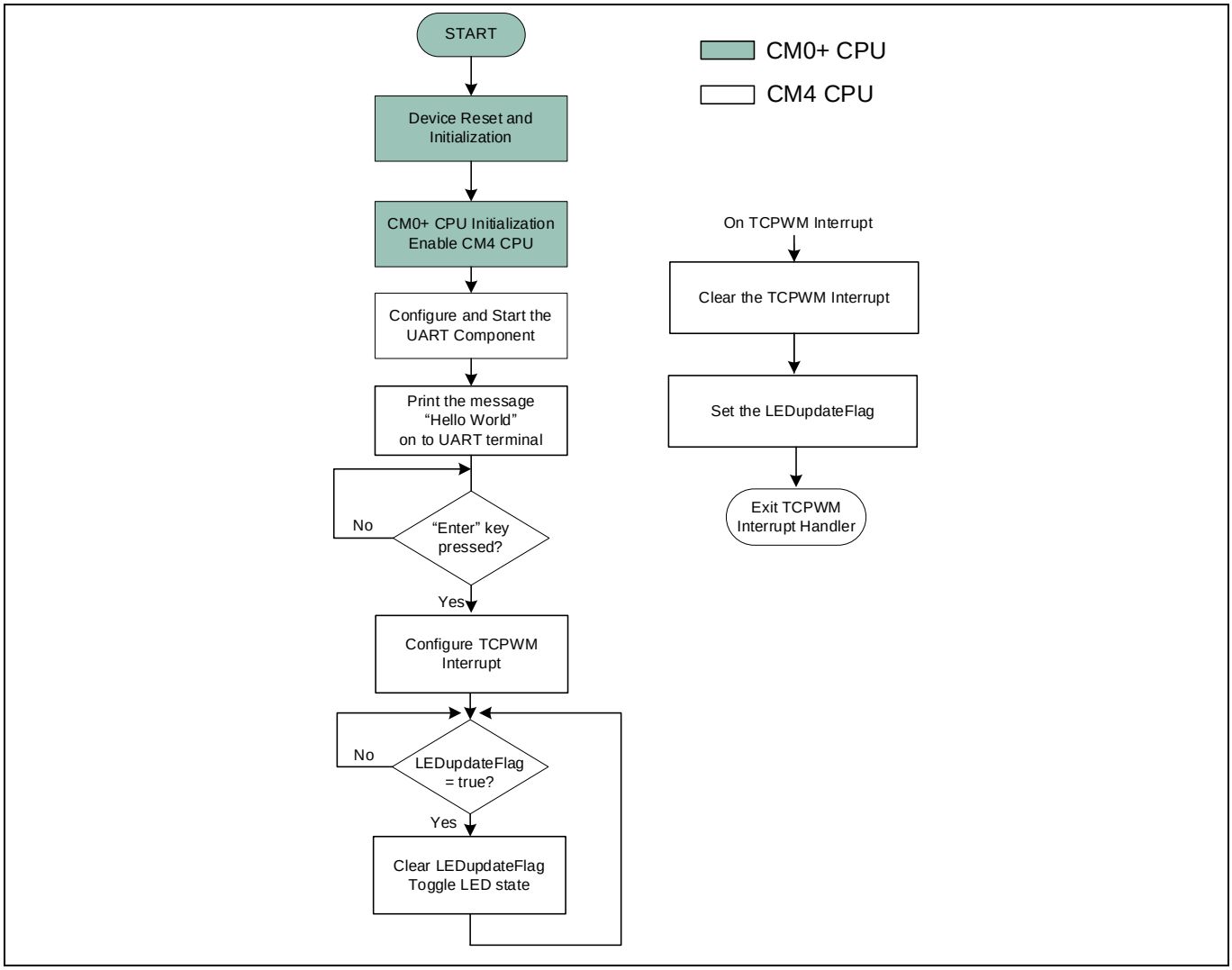

Figure 26 Firmware Flowchart

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

4.7 Part 5: Build the Project and Program the Device

This section shows how to program the PSoC 6 MCU device. If you are using a development kit with a built-in programmer (the CY8CKIT-062-WiFi-BT Pioneer Kit, for example), connect the board to your computer using the USB cable. If you are developing on your own hardware, you may need a hardware programmer/debugger; for example, **CY8CKIT-002 MiniProg3**.

If you are working from scratch and encounter errors, revisit prior steps to ensure that you accomplished all the required tasks. You can work to resolve errors or switch to the code example for these final steps.

1. **Select the debug target.**

   PSoC Creator can debug one CPU at a time.

   a) In PSoC Creator, choose **Debug** > **Select Debug Target**, as **Figure 27** shows.

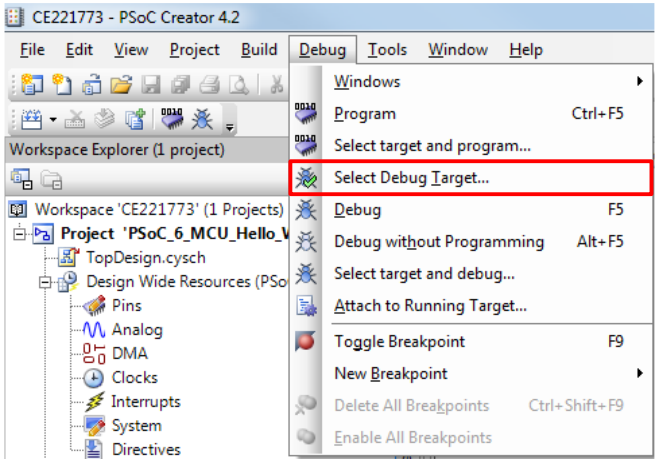

Figure 27 Selecting Debug Target

b) Connect to the board.

In the **Select Debug Target** dialog box, select the CM4 target, then click **OK/Connect**, as **Figure 28** shows.

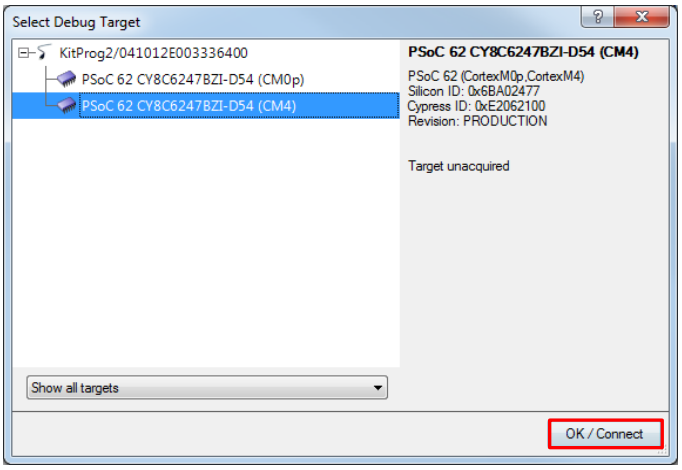

Figure 28 Connecting to a Device

Note: For programming the board, you can pick either target. The CPUs share the same memory space. Programming either CPU programs both CPUs. However, if you are debugging, this choice matters. The debugger will see only the CPU you connect to. These instructions do not use the debugger.

2. Program the board.

Choose **Debug** > **Program** to program the device with the project, as **Figure 29** shows.

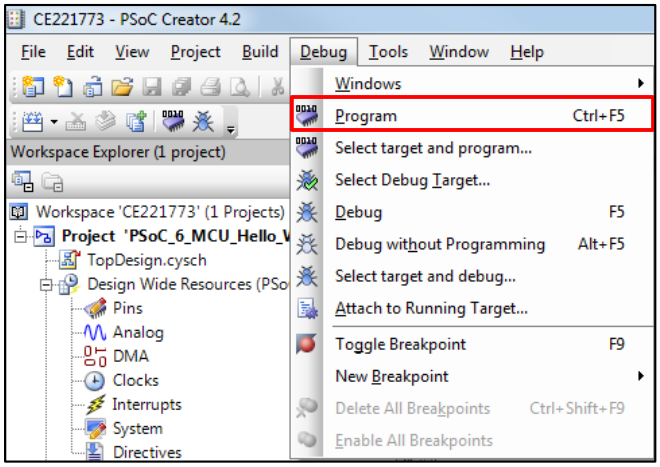

Figure 29 Programming the Device

You can view the programming status in the lower left corner of the PSoC Creator window, as **Figure 30** shows.

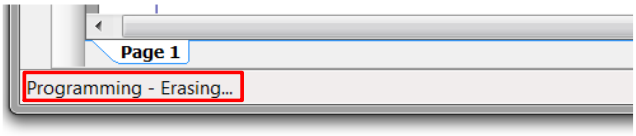

Figure 30 Programming Status

*Note: The **Debug** > **Debug** command also programs the board. If any code needs to be generated or rebuilt, that happens automatically when you issue a **Program** or **Debug** command. You can also debug without programming the board. However, these instructions do not use the debugger.*

Note: The KitProg2 firmware on the kit might require an update. See the respective kit user guide for step-wise instructions on updating the firmware.

4.8 Part 6: Test Your Design

This section describes how to test your design.

Follow the steps below to observe the output of your design. Note that the below steps use Tera Term as the UART terminal emulator to view the results. You can use any terminal of your choice to view the output.

1. **Select the serial port.**
   Launch Tera Term and select the KitProg2 USB-UART COM port as shown in **Figure 31**.

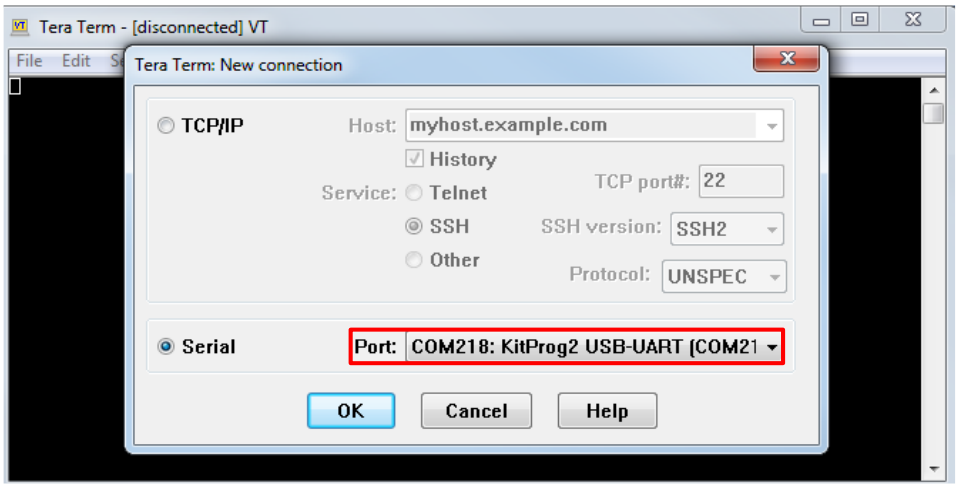

Figure 31 Selecting the KitProg2 USB-UART COM Port in Tera Term

2. Set the baud rate.

Set the baud rate to 115200 under **Setup** > **Serial port** as **Figure 32** shows.

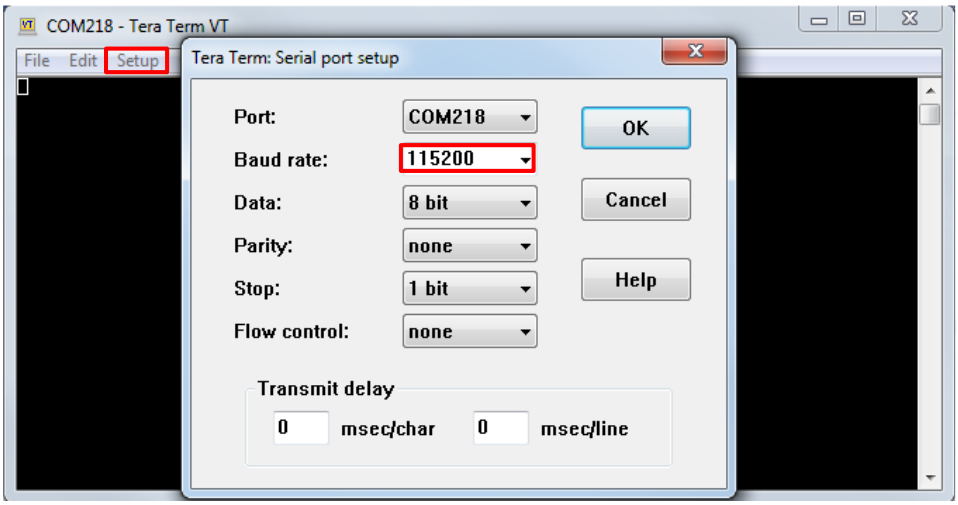

Figure 32 Configuring the Baud Rate in Tera Term

3. Reset the device.

Press the reset switch (SW1) on the Pioneer Kit. The following message appears on the terminal as **Figure 33** shows.

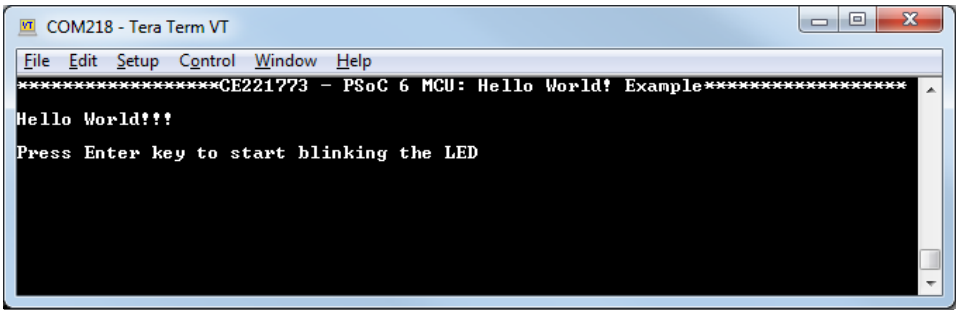

Figure 33 UART Message Printed from CM4 CPU

4. Enable the LED Blinking functionality.

Press the **Enter** key to start blinking the LED. When the LED starts blinking, the following message will be displayed on the UART terminal as shown in **Figure 34**.

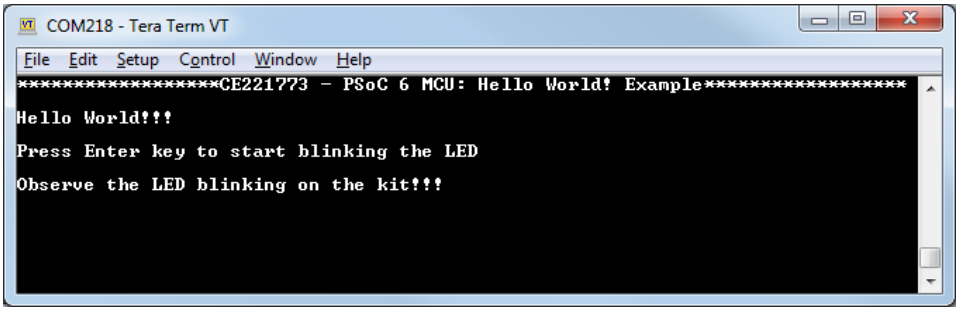

Figure 34 UART Message from CM4 CPU

5 Summary

This application note explored the PSoC 6 MCU device architecture and the associated development tools. PSoC 6 MCU is a truly programmable embedded system-on-chip with configurable analog and digital peripheral functions, memory, and a dual-CPU system on a single chip. The integrated features and low-power modes make PSoC 6 MCU an ideal choice for smart home, IoT gateways, and other related applications.

Related Application Notes and Code Examples

For a complete and updated list of PSoC 6 MCU code examples, please visit our **code examples web page**. For more PSoC 6 MCU-related documents, please visit our **PSoC 6 MCU** product web page.

Table 2 lists the system-level and general application notes that are recommended for the next steps in learning about PSoC 6 MCU and PSoC Creator.

Table 2 General and System-Level Application Notes

Document	Document Name
AN228571	Getting Started with PSoC 6 MCU on ModusToolbox
AN210781	Getting Started with PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity on PSoC Creator
AN218241	PSoC 6 MCU Hardware Design Considerations
AN219434	PSoC 6 MCU Importing Generated Code into an IDE
AN219528	PSoC 6 MCU Low-Power Modes and Power Reduction Techniques

Table 3 lists the application notes (AN) and code examples (CE) for specific peripherals and applications.

Table 3 Documents Related to PSoC 6 MCU Features

Document	Document Name
System Resources, CPU, and Interrupts	
AN215656	PSoC 6 MCU Dual-CPU System Design
AN217666	PSoC 6 MCU Interrupts
CE221773	PSoC 6 MCU Hello World Example
CE216795	PSoC 6 MCU Dual-Core Basics
CE216825	PSoC 6 MCU Real-Time Clock Basics
CE218129	PSoC 6 MCU Wake up from Hibernate Using Low-Power Comparator
CE218541	PSoC 6 MCU Fault-Handling Basics
CE218542	PSoC 6 Custom Tick Timer Using RTC Alarm Interrupt
CE218552	PSoC 6 MCU UART to Memory Buffer Using DMA
CE218964	PSoC 6 MCU RTC Daily Alarm
CE219339	PSoC 6 MCU MCWDT and RTC Interrupts (Dual Core)
CE219521	PSoC 6 MCU GPIO Interrupt
CE219881	PSoC 6 MCU Switching Power Modes
CE220060	PSoC 6 MCU Watchdog Timer
CE220061	PSoC 6 MCU Multi-Counter Watchdog Interrupts
CE220120	PSoC 6 MCU Blocking Mode Flash Write
CE220169	PSoC 6 MCU Periodic Interrupt Using TCPWM
GPIO	
CE219490	PSoC 6 Breathing LED Using SMART IO
CE219506	PSoC 6 Clock Buffer Using SMART IO
CE220263	PSoC 6 MCU GPIO Pins Example

Related Application Notes and Code Examples

Document	Document Name
CapSense	
AN92239	Proximity Sensing with CapSense
AN85951	PSoC 4 and PSoC 6 MCU CapSense Design Guide
Bootloader	
AN213924	PSoC 6 MCU Bootloader Software Development Kit (SDK) Guide
CE213903	PSoC 6 MCU Basic Bootloaders
Communications	
CE220541	PSoC 6 MCU SCB EzI2C
Audio	
CE218636	PSoC 6 MCU Inter-IC Sound (I2S) Example
CE219431	PSoC 6 MCU PDM-to-PCM Example
RTOS	
CE217911	PSoC 6 MCU FreeRTOS™ Example Project
Security	
CE220465	PSoC 6 MCU Cryptography – AES Demonstration
CE220511	PSoC 6 MCU Cryptography – SHA Demonstration

Glossary

This section lists the most commonly used terms that you might encounter while working with PSoC family of devices.

- **Component Customizer**: Simple GUI in PSoC Creator that is embedded in each Component. It is used to customize the Component parameters and is accessed by right-clicking a Component.
- **Components**: Components are used to integrate multiple ICs and system interfaces into one PSoC Component that is inherently connected to the MCU via the main system bus. For example, the Bluetooth LE Component creates Bluetooth Smart products in minutes. Similarly, you can use the Programmable Analog Components for sensors.
- **KitProg**: The KitProg is an onboard programmer/debugger with USB-I2C and USB-UART bridge functionality. The KitProg is integrated onto most PSoC development kits.
- **MiniProg3**/**MiniProg4**: Programming hardware for development that is used to program PSoC devices on your custom board or PSoC development 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.
- **PSoC**: A programmable, embedded design platform that includes a CPU, such as the 32-bit Arm Cortex-M0, with both analog and digital programmable blocks. It accelerates embedded system design with reliable, easy-to-use solutions, such as touch sensing, and enables low-power designs.
- **ModusToolbox**: An Eclipse based embedded design platform for IoT designers that provides a single, coherent, and familiar design experience combining the industry's most deployed Wi-Fi and Bluetooth technologies, and the lowest power, most flexible MCUs with best-in-class sensing.
- **PSoC Creator**: PSoC 3, PSoC 4, PSoC 5LP, PSoC 6 MCU, and PSoC 6 Bluetooth LE Integrated Design Environment (IDE) software that installs on your PC and allows concurrent hardware and firmware design of PSoC systems, or hardware design followed by export to other popular IDEs.
- **Peripheral Driver Library**: The Peripheral Driver Library (PDL) simplifies software development for the PSoC 6 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.
- **PSoC Programmer**: A flexible, integrated programming application for programming PSoC devices. PSoC Programmer is integrated with PSoC Creator to program PSoC 3, PSoC 4, PRoC, PSoC 5LP, PSoC 6 MCU, and PSoC 6 Bluetooth LE designs.
- **WICED**: WICED (Wireless Internet Connectivity for Embedded Devices) is a full-featured platform with proven Software Development Kits (SDKs) and turnkey hardware solutions from partners to readily enable Wi-Fi and Bluetooth connectivity in system design.

Revision history

Document version	Date of release	Description of changes
**	2018-03-28	New application note
*A	2018-10-04	Updated for ModusToolbox
*B	2018-10-31	Updated images
*C	2018-11-15	Updated for public release
*D	2019-02-19	Updated for ModusToolbox 1.1
*E	2019-10-14	Removed content for ModusToolbox 1.1
*F	2021-03-14	Updated to Infineon Template

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

Edition 2021-03-14
Published by
Infineon Technologies AG
81726 Munich, Germany

Do you have a question about this document?
Go to: www.cypress.com/support

Document reference
002-21774 Rev. *F

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.