CE223508 – PSoC 6 MCU Implementing BLE Multi-connection (4 Slaves)

Objective

This example demonstrates the implementation of multi-slave functionality of the PSoC® 6 MCU with BLE Connectivity (PSoC 6 BLE) device.

Overview

This code example implements Bluetooth Low Energy (BLE) multi-slave functionality that consists of the following:

- Device Information Service
- Health Thermometer Service
- Custom service for RGB LED with color and intensity control
- 128-bit long characteristic read write custom service
- A custom notification service

This code example also shows the connectivity between the PSoC 6 BLE, acting as a Peripheral and GATT Server, and four BLE enabled devices (a personal computer running the CySmart BLE Host Emulation tool or a mobile device running the CySmart mobile application) acting as a Central and GATT Client.

This code example assumes that you are familiar with the PSoC 6 MCU with BLE Connectivity device and the PSoC Creator™ integrated design environment (IDE). If you are new to PSoC 6 BLE, see the application note AN221774 – Getting Started with PSoC 6 MCU and AN210781 – Getting Started with PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity.

This code example uses FreeRTOS. See PSoC 6 101: Lesson 1-4 FreeRTOS training video to learn how to create a PSoC 6 FreeRTOS project with PSoC Creator. Visit the FreeRTOS website for documentation and API references of FreeRTOS. For the non-RTOS version of this code example see CE222004.

Requirements

Tool: PSoC Creator 4.2; Peripheral Driver Library (PDL) 3.0.3

Programming Language: C (Arm® GCC 5.4.2016-q2-update)

Associated Parts: All PSoC 6 MCU BLE devices with dual core parts

Related Hardware: CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit

Hardware Setup

This example uses the kit's default configuration. See the kit guide to ensure the kit is configured correctly.

Software Setup

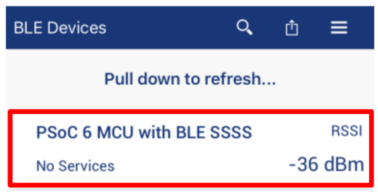
This code example requires CySmart application. Download and install either the CySmart Host Emulation Tool PC application or the CySmart app for iOS or Android. You can test the behavior with any of the two options, but the CySmart app is simpler. Scan one of the following QR codes from your mobile phone to download the CySmart app.

iOS	Android

Operation

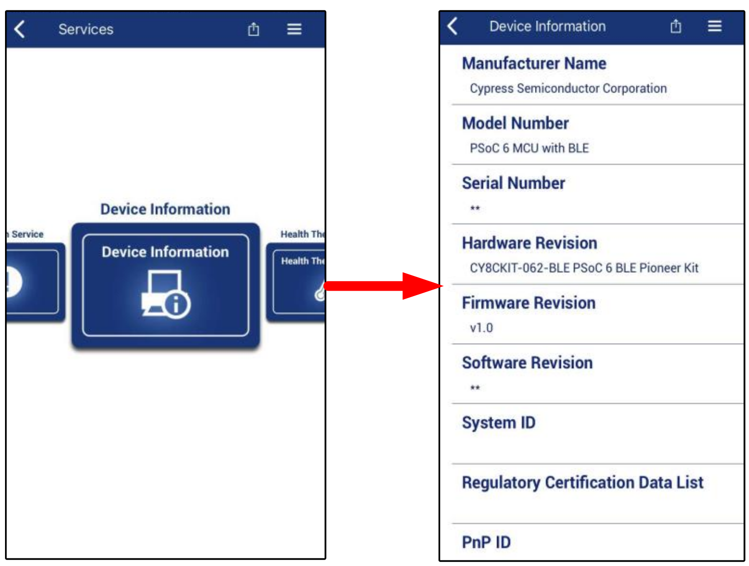
1. Plug the CY8CKIT-062-BLE kit board into your computer's USB port.
2. Build the project and program it into the PSoC 6 MCU device. Choose **Debug** > **Program**. For more information on device programming, see PSoC Creator Help. Flash for both CPUs is programmed in a single program operation. Do not replace the file *stdio_user.h* and *FreeRTOSConfig.h* files, if prompted by PSoC Creator.
3. Do the following to test using the CySmart mobile app:
   a. Press the reset switch on CY8CKIT-062-BLE to start the BLE advertisement. The advertisement LED (LED8) will start blinking to indicate that the BLE advertisement has started.
   b. Turn on Bluetooth on your Android or iOS device. Launch the CySmart app.
   c. Pull down the CySmart app home screen to start scanning for BLE Peripherals; your device appears in the CySmart app home screen as shown in Figure 1. Select your device to establish a BLE connection.

Figure 1. Device Selection



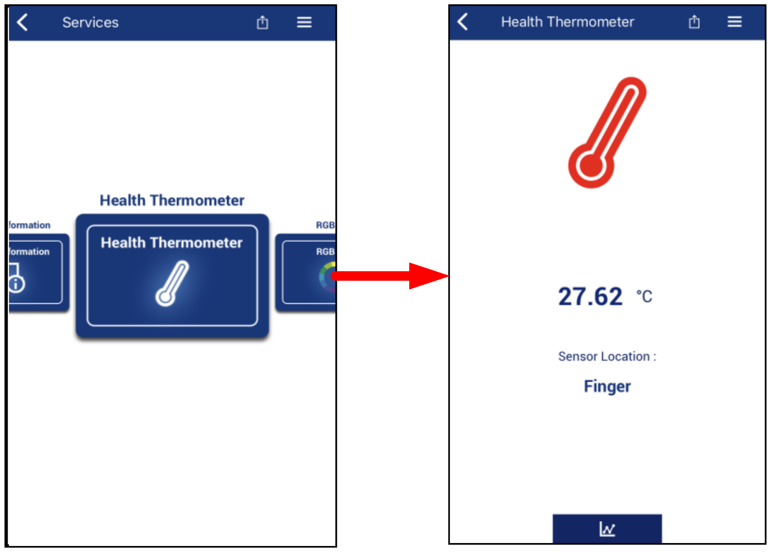
   d. To connect another device (mobile or PC), repeat steps b and c. Up to four Central devices can be connected to the PSoC 6 BLE device.
   e. On one or more of the connected devices, select the **Device Information** profile to get the manufacturer, vendor, or both information about the device, as Figure 2 shows.

Figure 2. Locating Device Information Using CySmart Mobile Application



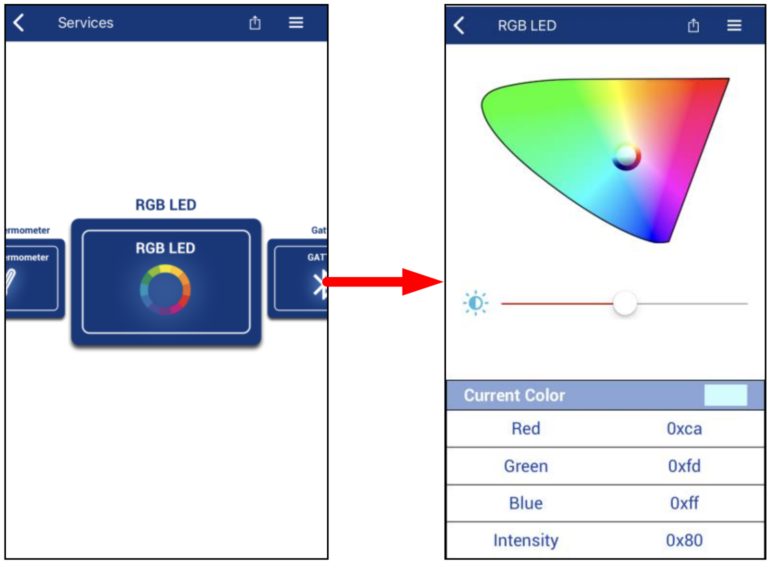
   f. Select the **Health Thermometer** profile to get the temperature information as shown in Figure 3.

Figure 3. Locating and Using Heath Thermometer Service Using CySmart Application



g. Select the **RGB LED** profile to change the color of the RGB LED (LED5) present in the kit as shown in Figure 4.

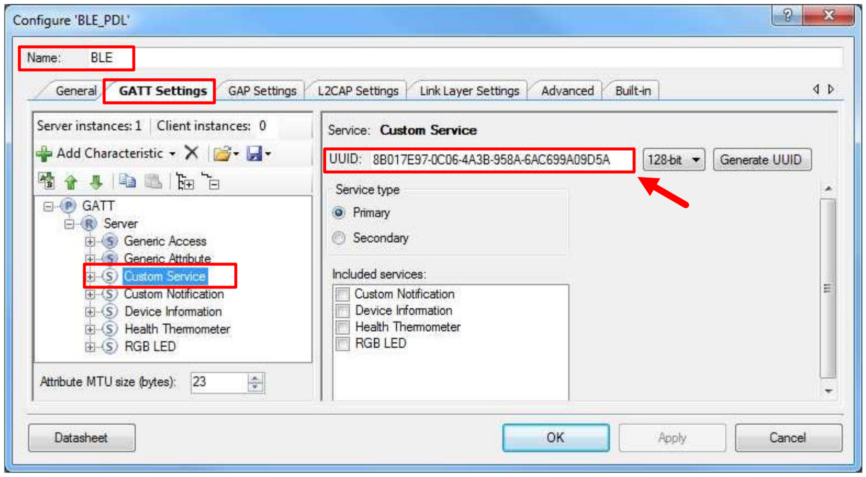
Figure 4. Locating and Using RGB LED Service Using CySmart Application



h. Custom Service and Custom Notification Service can be accessed through the GattDB profile. The service can be identified using its Universally Unique Identifier (UUID).

i. Locate the Custom Service characteristic (UUID 8B017E97-0C06-4A3B-958A-6AC699A09D5A). Using this service, a GAP Central device can read or write 128-bit data. You can get the UUID as Figure 5 shows.

Figure 5. Locating UUID



j. Locate the Custom Notification Service characteristic (UUID 4A8AA88D-C98C-411C-852E-DB06351DAF56). This service is used to notify the GAP central device that last modified any of the characteristics. The data payload contains two bytes of information.

XXYY → XX device id. (0 → Default, 1 to 4)
→ YY characteristics (0 → Default, 1 → Custom Service, 2 → RGB LED)

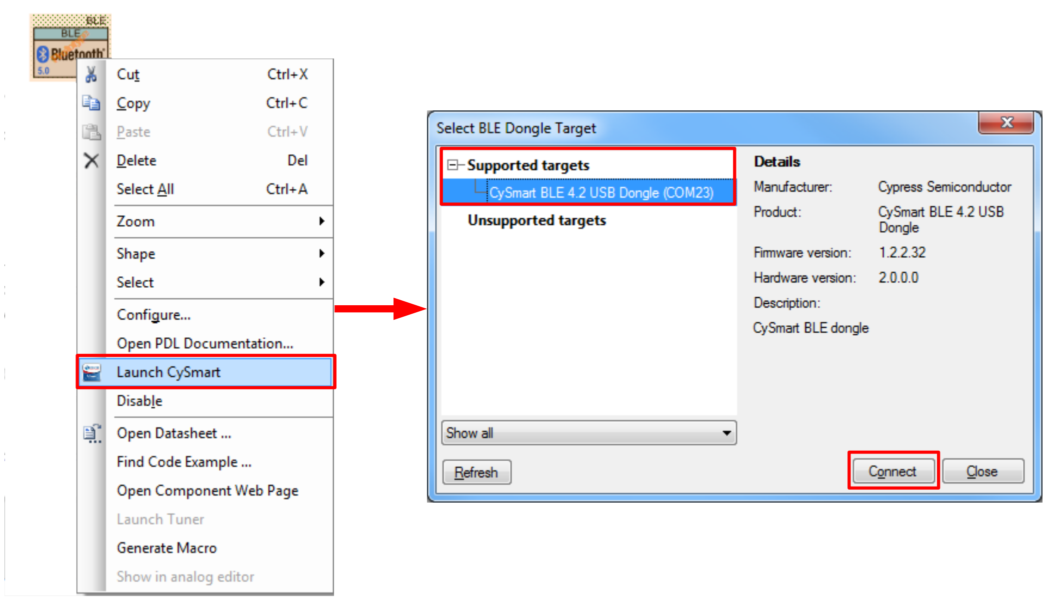
4. Do the following to test using the CySmart Host Emulation Tool:

a. Connect the BLE Dongle to your Windows PC. Wait for the driver installation to complete, if not done already.

b. Right-click the BLE Component and select **Launch CySmart** to launch the CySmart Host Emulation Tool. Alternatively, navigate to **Start** > **Programs** > **Cypress** and click **CySmart** to launch the tool.

CySmart automatically detects BLE Dongles connected to the PC. Click **Refresh** if the BLE Dongle does not appear in the **Select BLE Dongle Target** pop-up window. Click **Connect**, as shown in Figure 6.

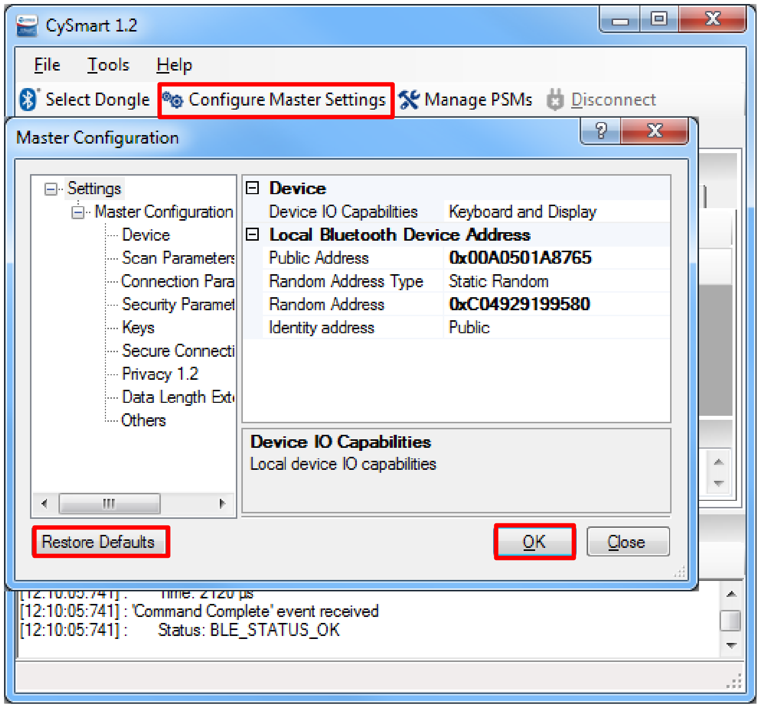
Figure 6. CySmart BLE Dongle Selection



Note: If the dongle firmware is outdated, you will be alerted with an appropriate message. You must upgrade the firmware before you can complete this step. Follow the instructions in the window to update the dongle firmware.

c. Select **Configure Master Settings** and then click **Restore Defaults**, as Figure 7 shows. Then, click **OK**.

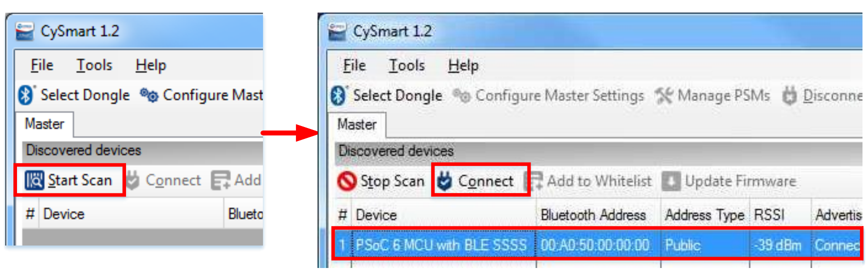
Figure 7. CySmart Master Settings Configuration



d. Press the reset switch on the Pioneer Kit to start the BLE advertisement if no device is connected or if the device is in Hibernate mode (red LED (LED9) is ON). Otherwise, skip this step.

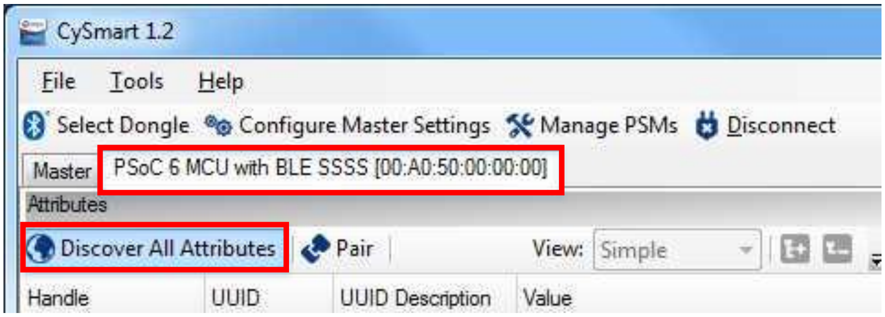
e. On the CySmart Host Emulation Tool, click **Start Scan**. Your device name (configured as 'PSoC 6 MCU with BLE SSSS') should appear in the Discovered devices list, as Figure 8 shows. Select the device and click **Connect** to establish a BLE connection between the CySmart Host Emulation Tool and your device.

Figure 8. CySmart Device Discovery and Connection



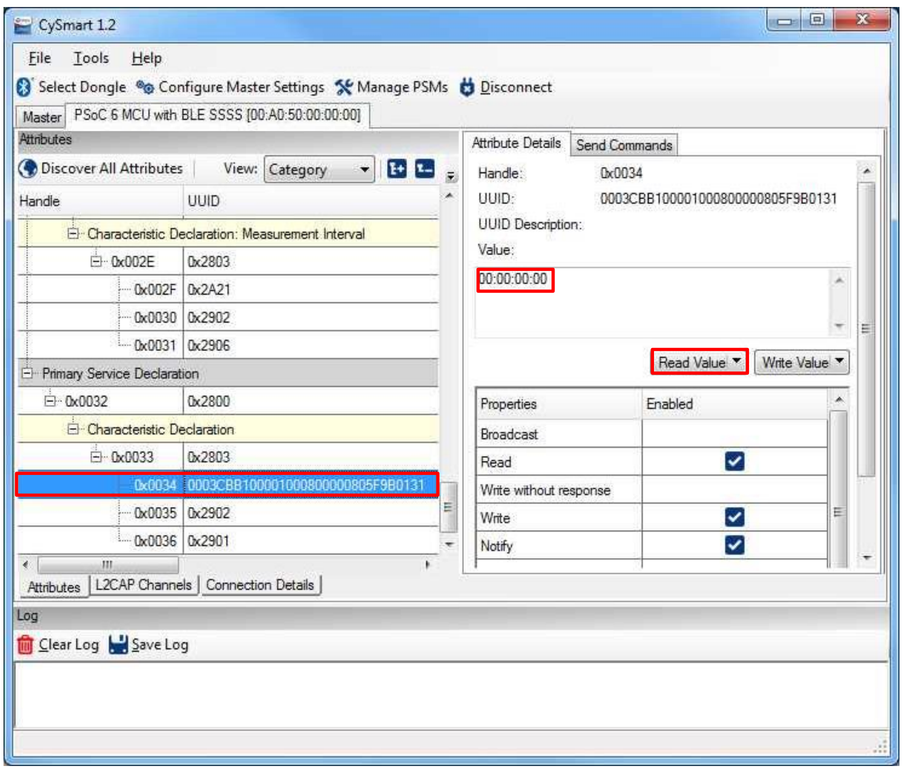
f. Once connected, switch to the **PSoC 6 MCU with BLE SSSS** device tab and click **Discover all Attributes** on your design from the CySmart Host Emulation Tool, as shown in Figure 9.

Figure 9. CySmart Attribute Discovery



g. Locate the RGB LED control characteristic (UUID 0x0003CBB1-0000-1000-8000-00805F9B0131). Click **Read Value** to read the existing 4-byte onboard RGB LED color information, as shown in Figure 10. The four bytes indicate red, green, blue, and the overall intensity, respectively. Modify the four bytes of data in the **Value** field to **FF:00:00:FF** and click **Write Value**. You will see the corresponding change in the color (Red) and intensity (full intensity) of the RGB LED on the Pioneer Board.

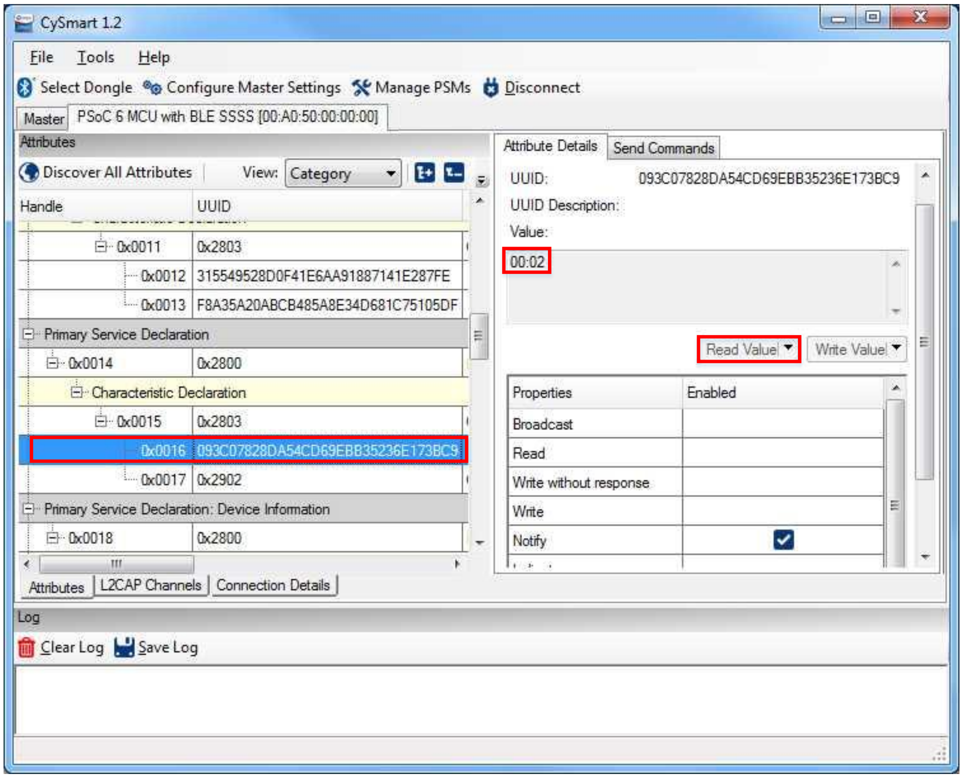
Figure 10. Locating RGB LED Characteristics



h. Locate the custom service control characteristic (UUID: 8B017E97-0C06-4A3B-958A-6AC699A09D5A). Click **Read Value** to read the 128-bit data from the GATT database. Click **Write Value** to write new data into the GATT database.

i. Locate the custom notification characteristic (UUID: 4A8AA88D-C98C-411C-852E-DB06351DAF56) as shown in Figure 11. This service notifies whenever any connected device modifies any data in GATT database. See step 3-j above for information about payload.

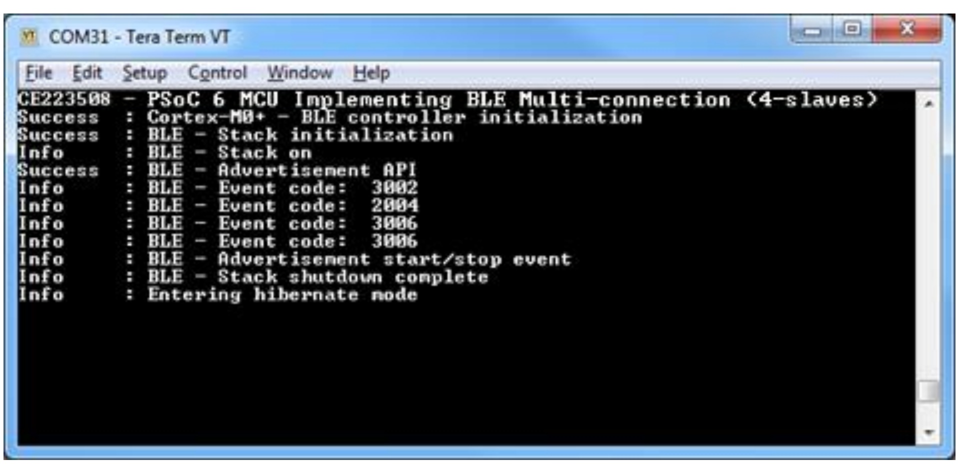
Figure 11. Locating Custom Notification Characteristics



j. If the device is in Hibernate mode, press the switch SW2 to restart advertisement.

5. Use the UART debug port to view verbose messages:
   a. The code example ships with the UART debug port disabled. To enable it, set the macro `UART_DEBUG_ENABLE` in *uart_debug.h* to `TRUE` and rebuild the code.
   b. Use a serial terminal application and connect to the **KitProg2 USB-UART** COM port. Configure the application to access the COM port at 115200 bps baud rate.
   c. Program the board. The debug messages will appear in the terminal window as shown in Figure 12.

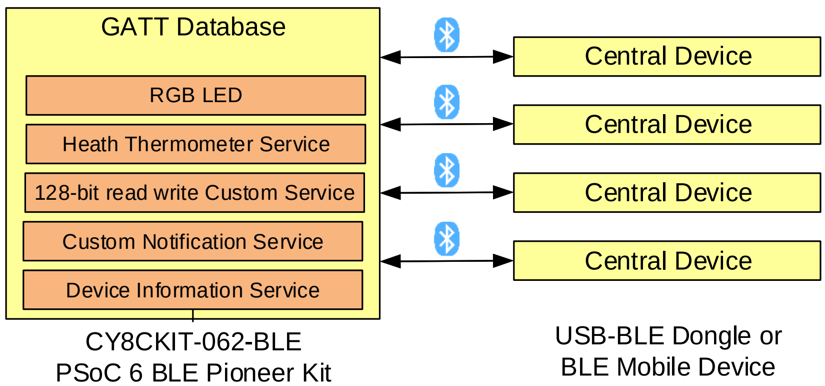
Figure 12. Debug Messages on COM Port



Design and Implementation

The PSoC 6 MCU device is a BLE GATT Server. It can connect to as many as four GAP Central devices, as shown in Figure 13. All connected GAP Central devices can access the GATT database.

Figure 13. Service Relationship



This code example features the following:

- BLE connectivity:
  - Advertisement and connection with four central devices
  - Five services (RGB LED, Health Thermometer Service, Device Information, read write 128-bit long Custom Service, and Custom Notification Service)
  - Data transfer over BLE using notifications, read, and write
- RGB LED color and intensity control using configurable digital blocks of PSoC 6 MCU.
- ADC scans two differential channels and averages multiple samples without the need for CPU intervention for accurate temperature measurement from a thermistor circuit.
- Device information service gives the manufacturer, vendor, or both information about the device.
- Custom service is used to transfer 128-bit data.
- Custom notification is used to send a notification to all connected devices about any changes. It sends two-byte data, the first is to notify the device that modified the data and the second byte is to notify the characteristic data that has been modified.
- Low-power operation using the Deep Sleep mode with MCWDT and GPIO interrupts.
- The kit orange (LED8) and red (LED9) LEDs are used to show the status:
  - If the device is in Hibernate mode the red LED will be ON.
  - If BLE is advertising, the orange LED will blink.

Figure 14 to Figure 18 show the TopDesign schematic of this code example.

Figure 14. TopDesign Schematic: BLE and GPIO

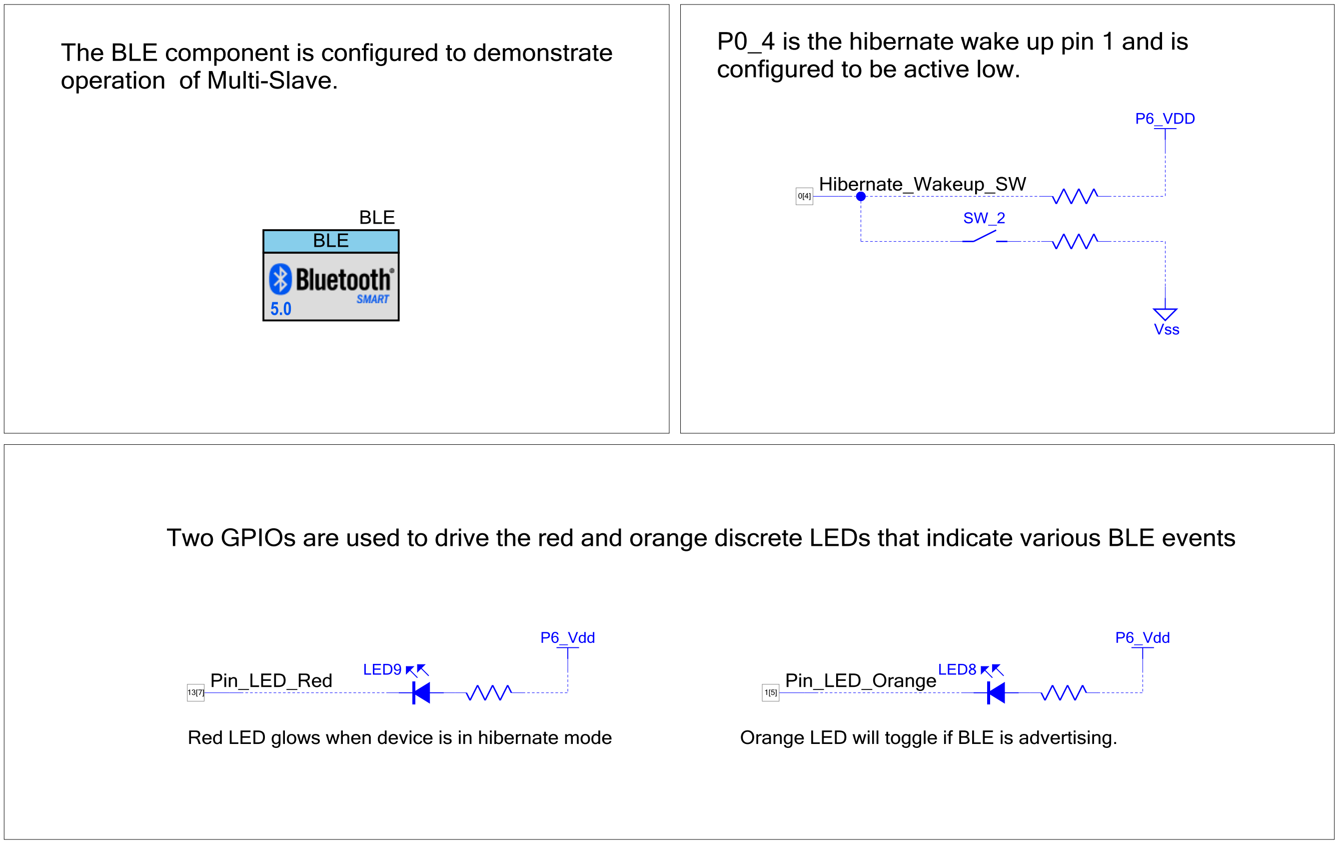


Figure 15. TopDesign Schematic: UART

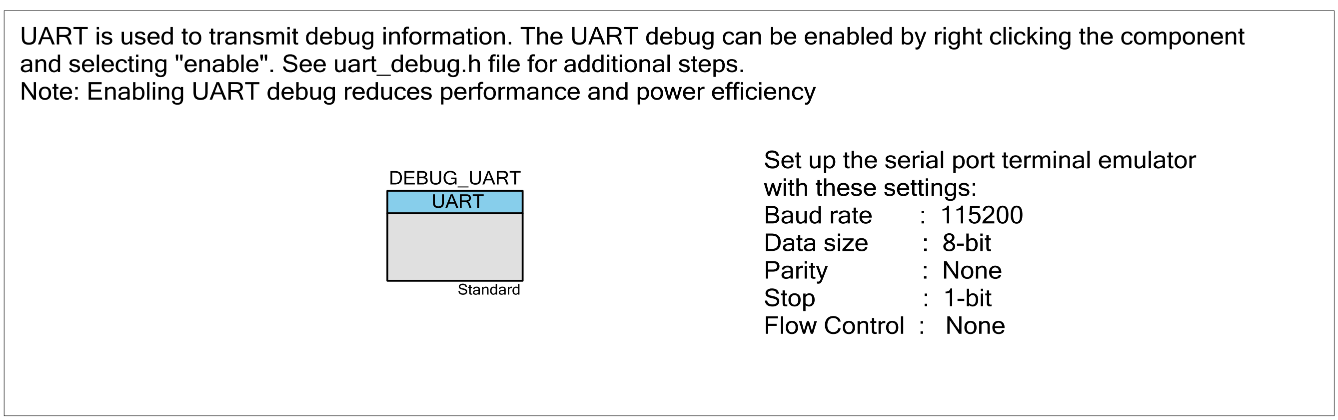


Figure 16. TopDesign Schematic: ADC and GPIO for Thermometer

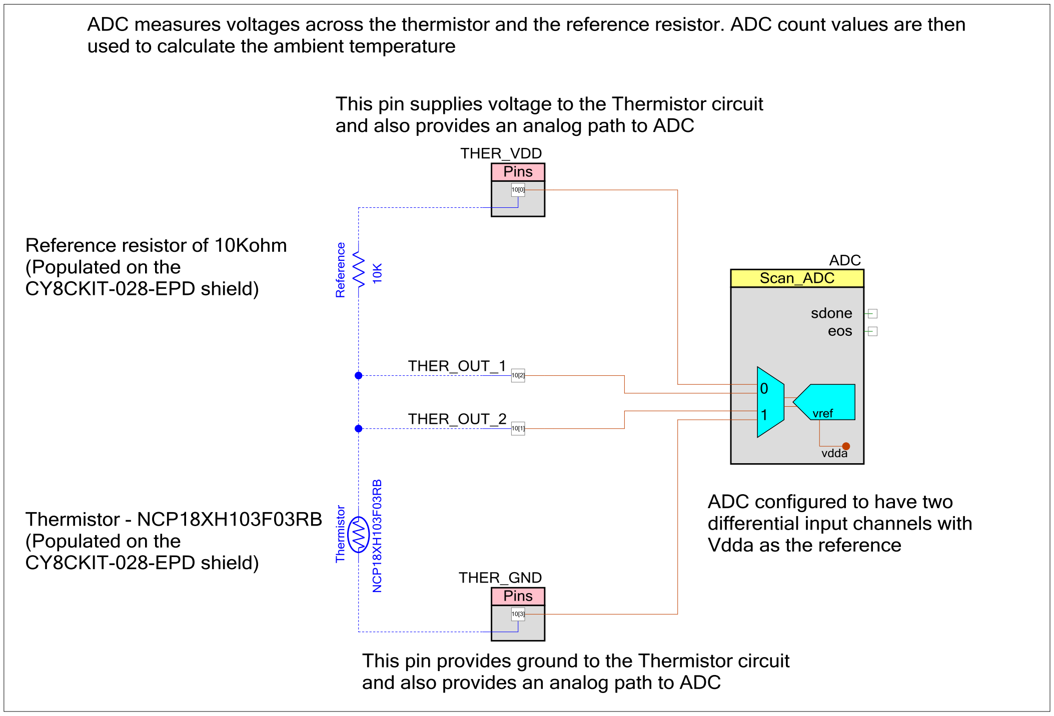


Figure 17. TopDesign Schematic: PWMs and GPIOs

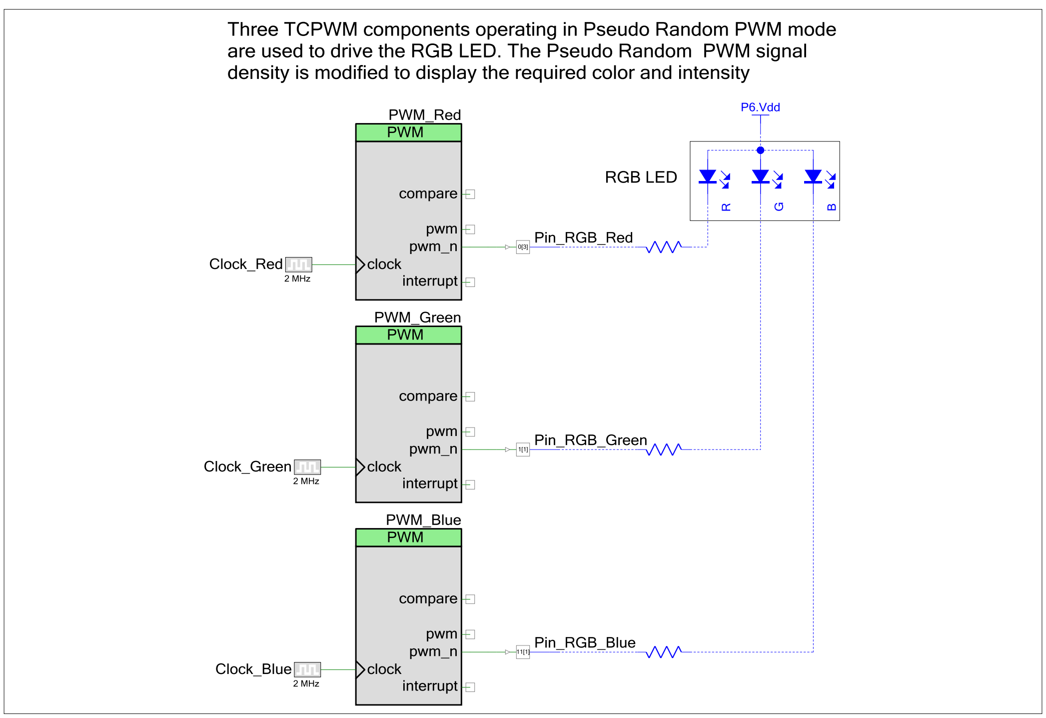
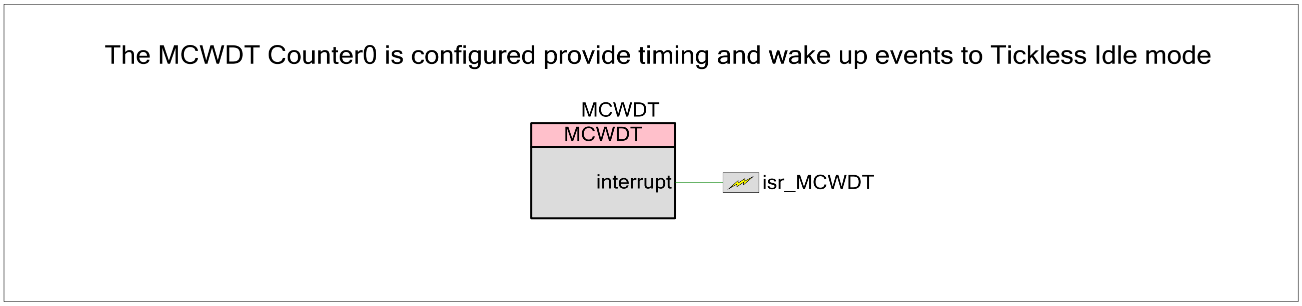


Figure 18. TopDesign Schematic: MCWDT

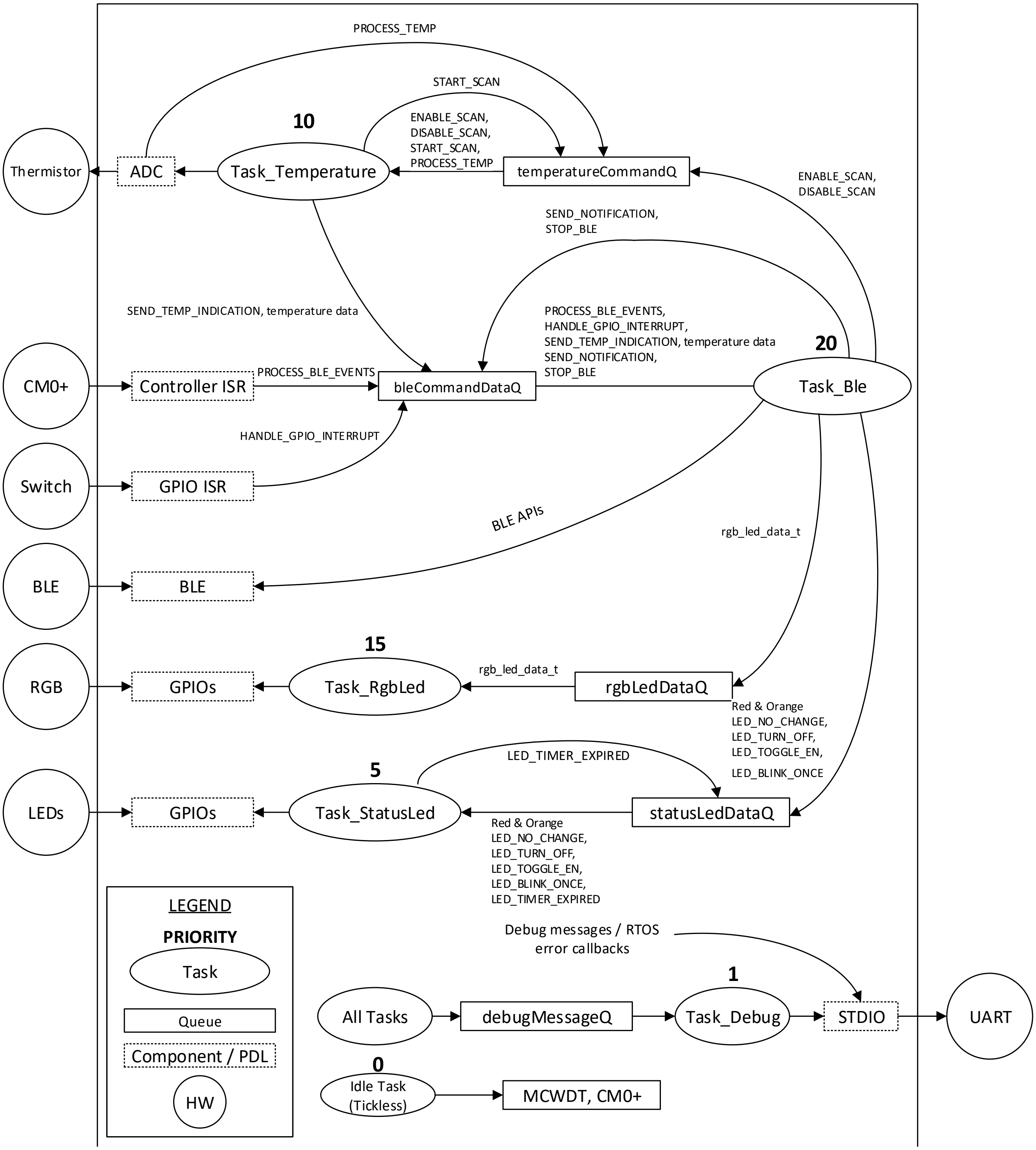


The code example consists of the following files:

- *FreeRTOSConfig.h* contains the FreeRTOS settings and configuration. Non-default settings are explained with in-line comments.
- *main_cm0p.c* contains functions that start the BLE controller, enable the CM4 core, enable UART Component, and BLE stack events
- *main_cm4.c* contains the main function, which is the entry point and execution of the firmware application. The main function sets up user tasks and then starts the RTOS scheduler.
- *ble_task.c/.h* contain the task and associated functions that handle BLE communication and operation.
- *ble_custom_service_config.h* contains the macros and datatypes used for the three custom BLE services.
- *rgb_led_task.c/.h* contain the task that initialize and control the RGB LED and intensity.
- status_led_task.c/h – contain the task that controls status LED indications.
- *uart_debug.c/h* contain the task and functions that enable UART based debug message printing.
- *temperature_task.c/h* contain functions that measure ambient temperature.
- *tickless_idle.c/h* contains functions and RTOS hooks used for Tickless idle mode.

Figure 19 shows the firmware flow of this code example.

Figure 19. RTOS Firmware Flow



Components and Settings

Table 1 lists the PSoC Creator Components used in this example, how they are used in the design, and the non-default settings required so they function as intended.

Table 1. PSoC Creator Components

Component	Instance Name	Purpose	Non-default Settings
Bluetooth Low Energy (BLE)	BLE	The BLE Component is configured to act as four Peripherals and GATT Servers.	See the Parameter Settings section
Digital Input Pin	Hibernate_Wakeup_SW	This pin is used to generate interrupts when the user button (**SW2**) is pressed.	**[General tab]** Uncheck HW connection Drive mode: Resistive Pull Up
Analog Pin	THER_OUT_1, THER_OUT_2	These GPIOs connect the thermistor circuit output to the ADC input.	Default
Analog and Digital Output Pin	THER_VDD	These GPIOs provide power to the thermistor circuit and provides an analog path to the ADC.	**[General tab]** Check Analog Check Digital Output Uncheck HW connection Initial drive state: High
Analog and Digital Output Pin	THER_GND	These GPIOs provide ground to the thermistor circuit and provides an analog path to the ADC.	**[General tab]** Check Analog Check Digital Output Uncheck HW connection Initial drive state: Low
Digital Output pin	Pin_LED_Red Pin_LED_Orange Pin_RGB_Red Pin_RGB_Blue Pin_RGB_Green	These GPIOs are configured as firmware-controlled digital output pins that control LEDs.	**[General tab]** Uncheck HW connection Drive mode: Strong Drive
TCPWM	PWMPR_Red PWMPR_Blue PWMPR_Green	These PWMs are used to control the brightness of the LEDs.	**[General tab]** PWM Mode: PWM Pseudo Random
Scanning SAR ADC	ADC	The ADC measures the voltages across a thermistor and a series reference resistor using two differential channels.	See the Parameter Settings section
MCWDT	MCWDT	The MCWDT Counter0 is configured provide timing and wake up events to Tickless Idle mode.	See the Parameter Settings section
SysInt	MCWDT_isr	This Component is configured to extract interrupts from MCWDT.	Default
UART (SCB)	DEBUG_UART	This Component is used to print messages on a terminal program.	Default

For information on the hardware resources used by a Component, see the respective Component datasheet.

Parameter Settings

For more information on Component configuration options, see the respective Component datasheets. Figure 20 through Figure 25 show the modified settings for the BLE Component.

Figure 20. BLE: Protocol Configuration

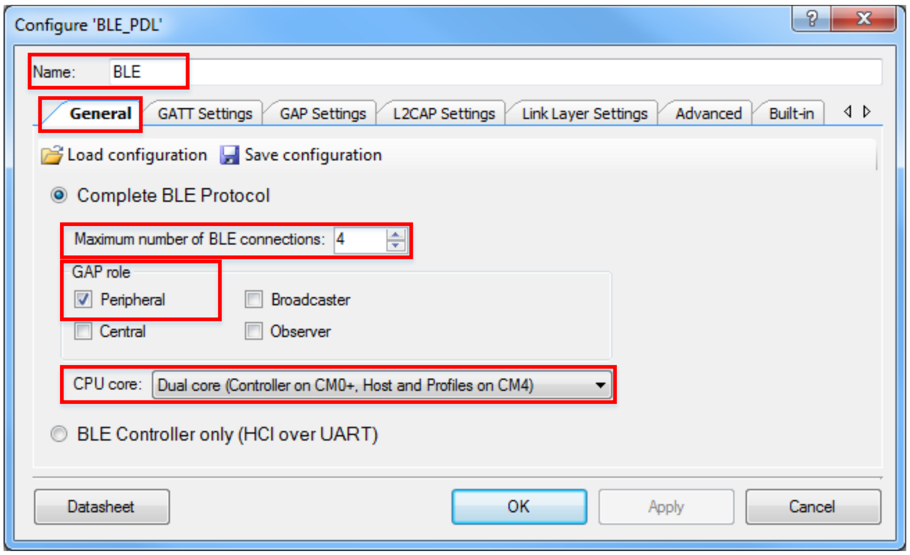


Figure 21. BLE: Adding Profiles

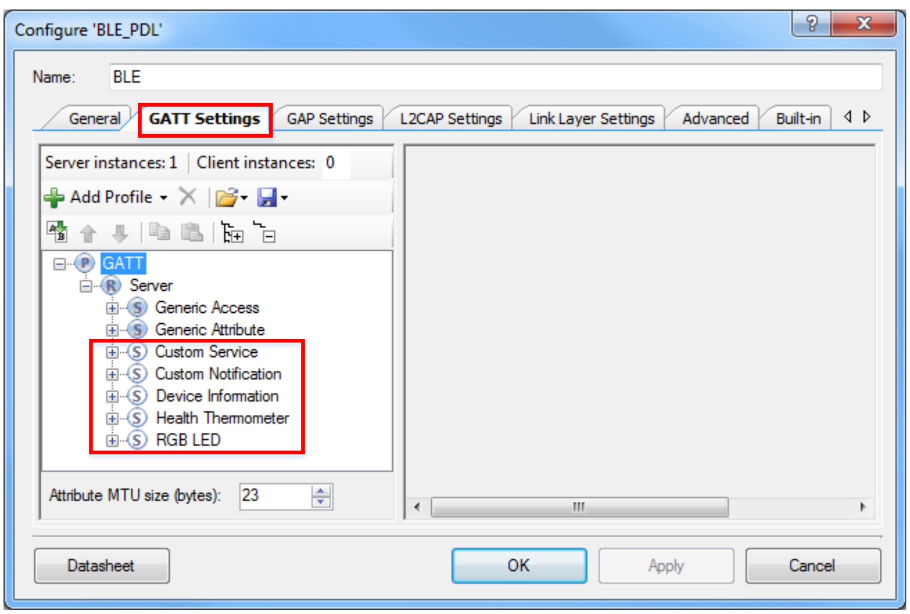


Figure 22. BLE: Device Configuration

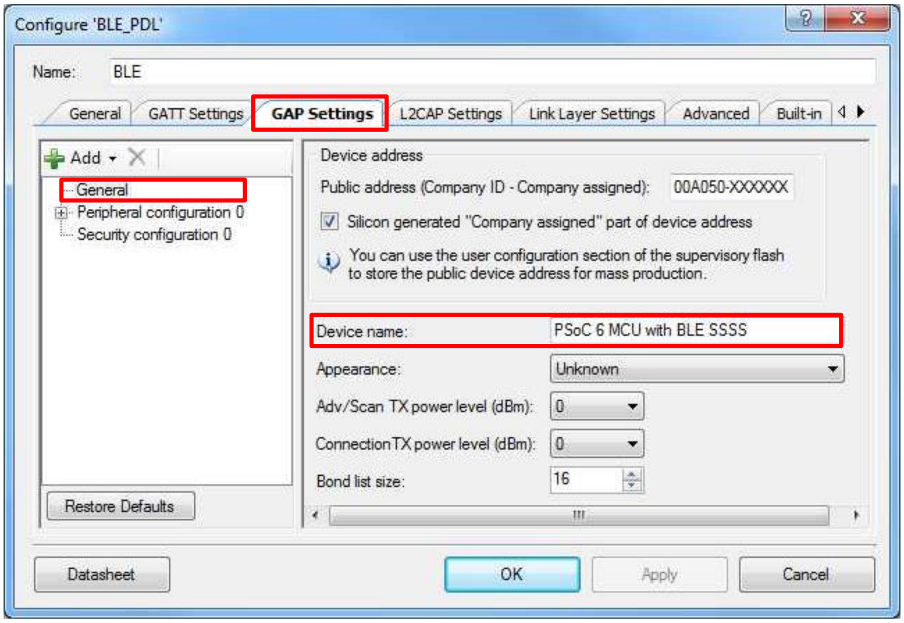


Figure 23. BLE: Advertisement Settings

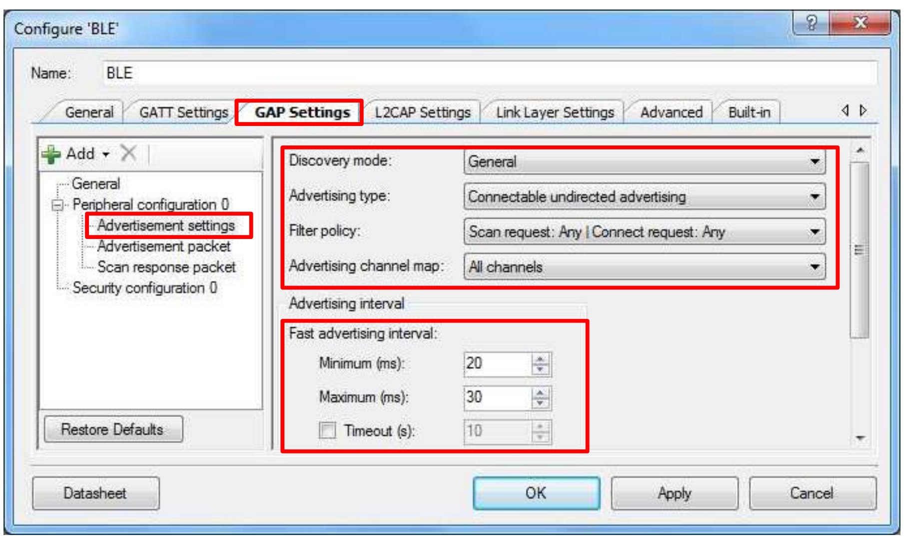


Figure 24. BLE: Advertisement Packet Settings

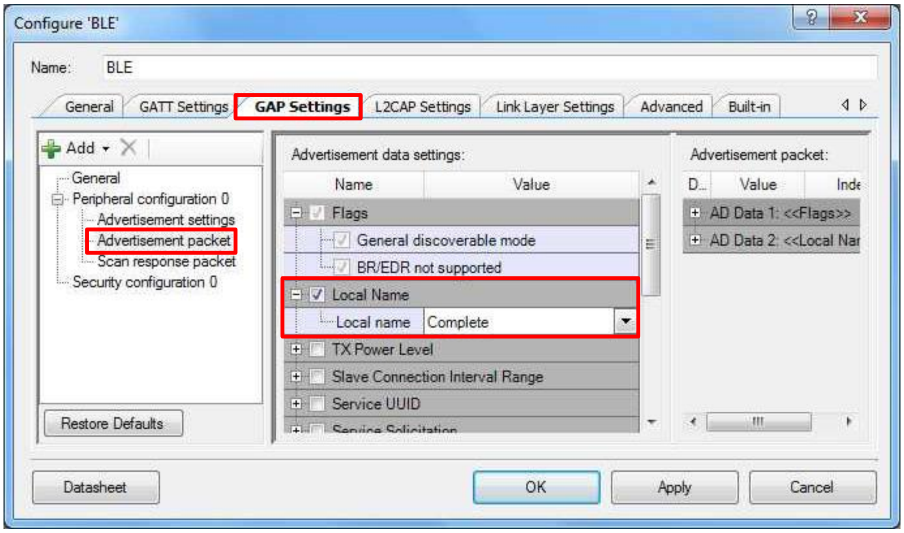
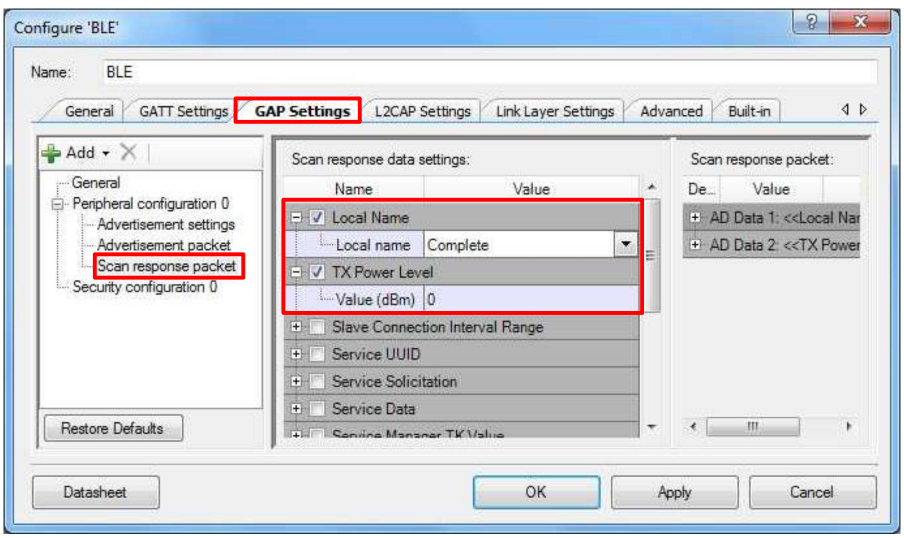


Figure 25. BLE: Response Packet Settings



MCWDT Counter 0 is configured to generate an interrupt every 250 ms as shown in Figure 26. Also, note that only Counter 0 is enabled.

Figure 26. MCWDT Component Configuration

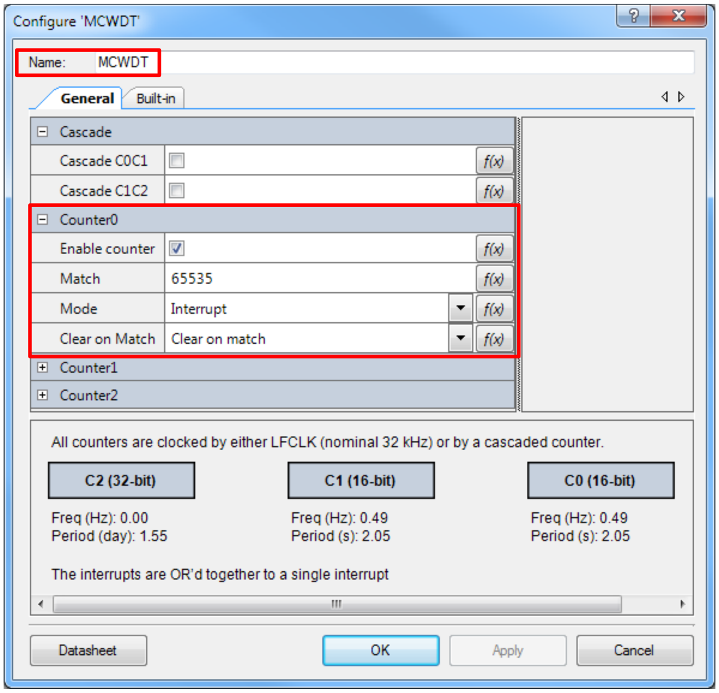


Figure 27 shows the Component settings for ADC.

Figure 27. ADC Component Settings

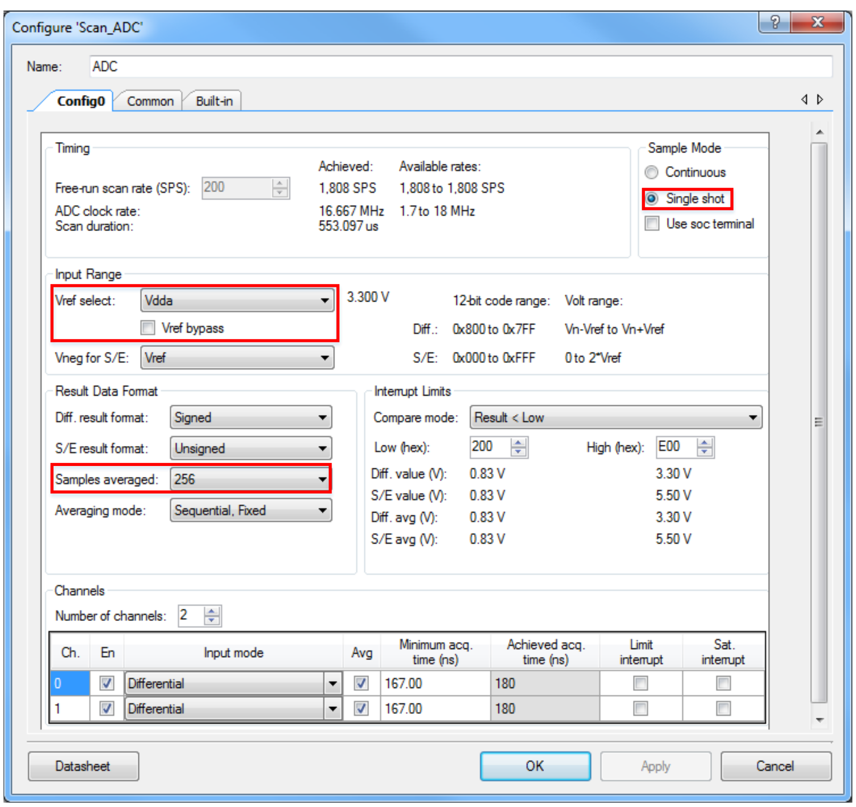


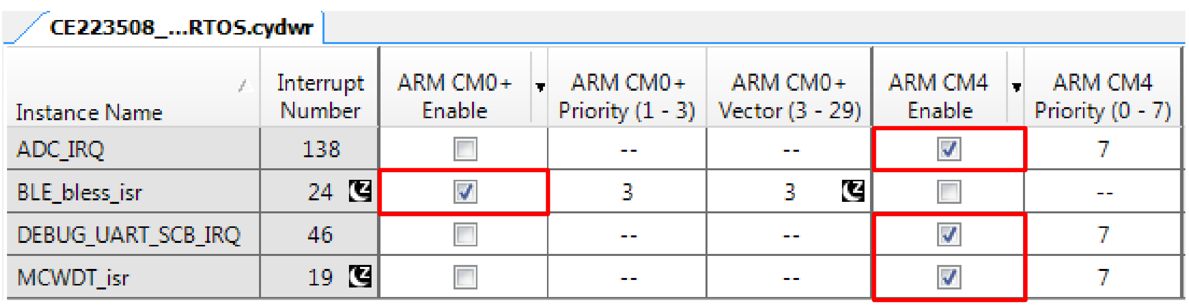
Table 2 shows the pin assignment for the project done through the **Pins** tab in the **Design Wide Resources** window. These assignments are compatible with CY8CKIT-062-BLE.

Table 2. DWR Pin Assignment Table

Instance Name	Pin
Hibernate_Wakeup_SW	P0[4]
Pin_LED_Orange	P1[5]
Pin_LED_Red	P13[7]
Pin_RGB_Blue	P11[1]
Pin_RGB_Green	P1[1]
Pin_RGB_Red	P0[3]
THER_GND	P10[3]
THER_VDD	P10[0]
THER_OUT_1	P10[2]
THER_OUT_2	P10[1]
UART_DEBUG:rx	P5[0]
UART_DEBUG:tx	P5[1]

Figure 28 shows the interrupt configuration for the project.

Figure 28. System Interrupt Configuration



CE223508_...RTOS.cydwr

Instance Name	Interrupt Number	ARM CM0+ Enable	ARM CM0+ Priority (1 - 3)	ARM CM0+ Vector (3 - 29)	ARM CM4 Enable	ARM CM4 Priority (0 - 7)
ADC_IRQ	138	☐	--	--	☑	7
BLE_bless_isr	24	☑	3	3	☐	--
DEBUG_UART_SCB_IRQ	46	☐	--	--	☑	7
MCWDT_isr	19	☐	--	--	☑	7

Related Documents

Application Notes	
AN210781 – Getting Started with PSoC 6 MCU with BLE Connectivity	Describes PSoC 6 MCU with BLE Connectivity devices and how to build your first PSoC Creator project
AN221774 – Getting Started with PSoC 6 MCU	Describes PSoC 6 MCU devices and how to build your first PSoC Creator project
AN215656 – PSoC 6 MCU Dual-CPU System Design	Describes the dual-CPU architecture in PSoC 6 MCU, and shows how to build a simple dual-CPU design
AN219434 – Importing PSoC Creator Code into an IDE for a PSoC 6 MCU Project	Describes how to import the code generated by PSoC Creator into your preferred IDE
Code Examples	
CE215118 - BLE Multi-Master Single Slave with PSoC 6 MCU with BLE Connectivity	
CE220167 - PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity: BLE with User Interface	
CE222004 - PSoC 6 MCU with BLE Multi- Master Multi-Slave: SSSS Function	
PSoC Creator Component Datasheets	
Bluetooth Low Energy	Facilitates designing applications requiring BLE connectivity.
MCWDT	Provides MCWDT settings
Pins	Supports connection of hardware resources to physical pins
SysInt	Provides SysInt component settings
UART	Provides asynchronous serial communications
Pins	Supports connection of hardware resources to physical pins
Timer Counter (TCPWM)	Supports fixed-function Timer/Counter implementation
Clock	Supports local clock generation
Interrupt	Supports generating interrupts from hardware signals
Device Documentation	
PSoC® 6 MCU: PSoC 63 with BLE Datasheet	PSoC® 6 MCU: PSoC 63 with BLE Architecture Technical Reference Manual
Development Kit (DVK) Documentation	
CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit	

Document History

Document Title: CE223508 – PSoC 6 MCU Implementing BLE Multi-connection (4 Slaves)

Document Number: 002-23508

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	6213104	AJYA	06/20/2018	New code example
*A	6300414	AJYA	09/06/2018	Minor document update

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

Arm® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6 MCU

Cypress Developer Community

Community | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709