Chapter 8: Low Power

The ModusToolbox™ level 2 PSoC class has a chapter that goes into details of achieving low power with PSoC™ 6 and FreeRTOS. Those concepts still apply to PSoC™ 6 + 43xxx Bluetooth®, but there are some additional things to be aware of regarding low power in the 43xxx Bluetooth® device. Those topics are covered in this chapter.

Table of contents

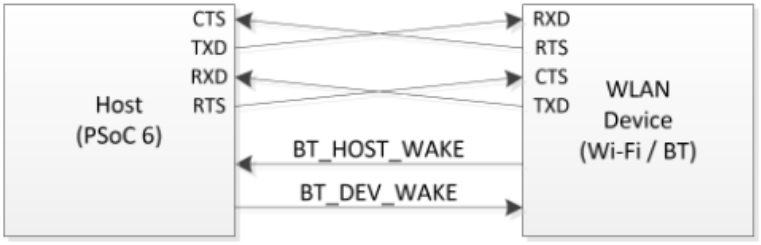
Document conventions

Convention	Usage	Example
`Courier New`	Displays code and text commands	`CY_ISR_PROTO(MyISR);` `make build`
Italics	Displays file names and paths	*sourcefile.hex*
[**bracketed, bold**]	Displays keyboard commands in procedures	[**Enter**] or [**Ctrl**] [**C**]
Menu > Selection	Represents menu paths	**File > New Project > Clone**
Bold	Displays GUI commands, menu paths and selections, and icon names in procedures	Click the **Debugger** icon, and then click **Next**.

8.1 Hardware connections

As you learned previously, the PSoC 6 communicates with the Bluetooth portion of the connectivity device using a UART interface. In addition to the UART interface, there are two pins dedicated for use in low power. These pins allow the Bluetooth LE device to wake the PSoC 6 host and vice versa. In this way, either device can go into a low power mode whenever its application allows, and the other device can wake it up when it is needed to handle specific activity.

Refer to the LPA Library Guide Part 3 for additional information.



- UART: (CTS / TXD / RXD / RTS)
- BT_HOST_WAKE (host wake): MCU input pin which can wake the MCU with interrupt.
- BT_DEV_WAKE (device wake): an output MCU host pin which is connected as an input Bluetooth device pin which interrupts the Bluetooth device when set in active state.

8.2 Low power assistant for Bluetooth®

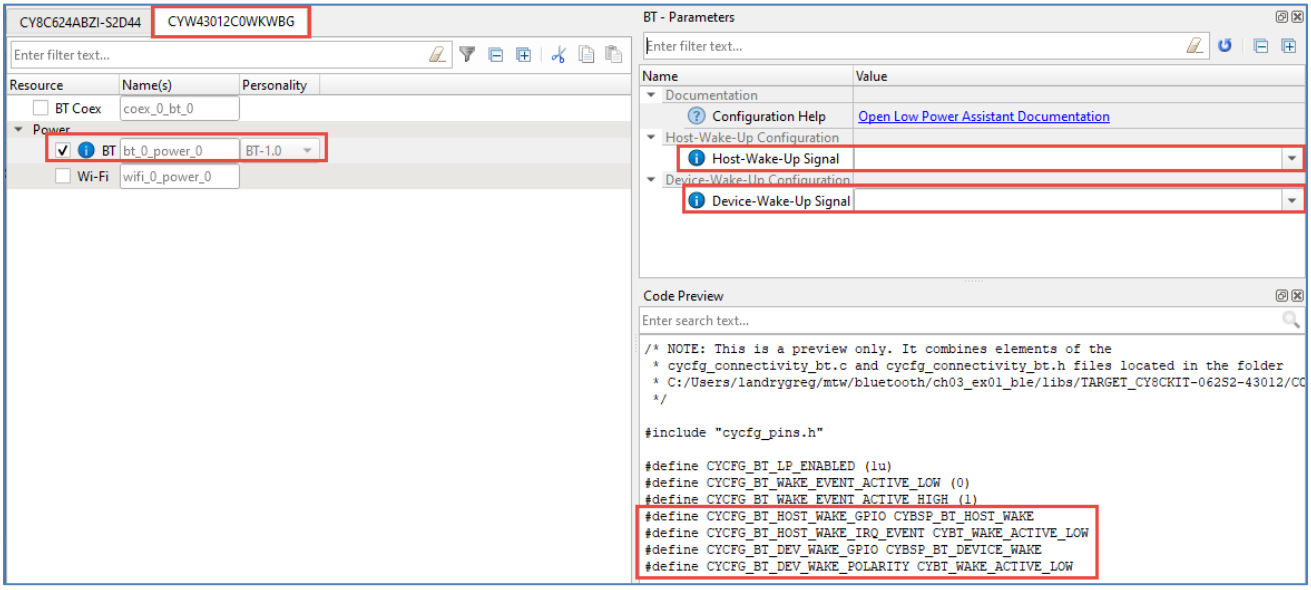
In the ModusToolbox™ level 2 PSoC™ class you learned about achieving low power on the PSOC™. However, it is also important to reduce power on the connectivity device. There is a low power assistant that is used to achieve low power on both the Wi-Fi and Bluetooth® interfaces but we will only cover Bluetooth® here.

Low power is enabled by default in the BSP so you don't have to do anything to get the best system power, but it is still useful to understand what is happening in the system.

Note: *There is a low power assistant library that provides an API for low power functions. However, this library is only needed for Wi-Fi. When implementing low power for Bluetooth®, we still call it the low power assistant, but there is no additional library required.*

8.2.1 Device Configurator

To access the Bluetooth® low power assistant settings in the device configurator, switch to the tab for the connectivity device, then make sure the box is checked for **Power > BT**.



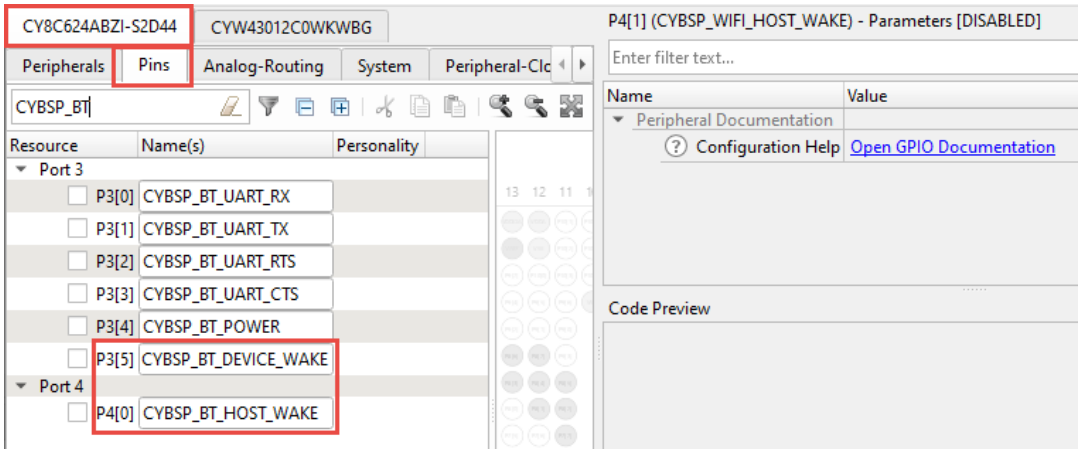
The first thing you will notice is that the Host-Wake-Up Signal and Device-Wake-Up Signal boxes are empty. That seems like a problem but in this case, it is actually OK because these pins have default names of `CYBSP_BT_HOST_WAKE` and `CYBSP_BT_DEVICE_WAKE`. There are also default values for the interrupt priorities for those pins. You can see the default names in the code preview that is shown in the low power assistant.

Note: *If you want to know where those default names are defined, they are in the personality file for the PDL library that is loaded into the configurator. It can be found at: mtb_shared/mtb-pdl-cat1/<version>/personalities/peripheral/connectivity_bt_intrinsic-<version>.cypersonality.*

If you disable the **Power > BT** settings, the same macros are generated, but the values are updated so that low power is disabled and the pins are not assigned or configured. For example, with low power disabled, the code will look like this:

```
#define CYCFG_BT_LP_ENABLED 0
…
#define CYCFG_BT_HOST_WAKE_GPIO CYHAL_NC_PIN_VALUE
…
```

The BSP configuration for the MCU assigns the proper names for the desired pins as you can see on the **MCU > Pins** tab. Therefore, you don't need to enable the pins or specify them in the **Power > BT** settings.



Note: *The drive configuration for the GPIOs is done by the bluetooth-freertos library using the HAL.*

Note: *The same default naming is used for the UART pins between the MCU and connectivity device (e.g. `CYBSP_BT_UART_RX`, `CYBSP_BT_UART_TX`, etc.) as you can see above. The drive modes for those pins are also setup by the bluetooth-freertos library.*

You can use the boxes in the low power assistant settings if you want to override the pin assignments from the default values, but it is easier to just assign the default names to the correct pins in the **MCU > Pins** tab. If, for some reason you want to specify the pins in the low power settings, you must first enable those pins on the **MCU > Pins** tab, configure them properly and then select them (and their polarities) in the drop-downs on the low power assistant settings.

8.2.2 BSP

Now that the pins are defined and configured, how are they used? The answer is that they are used by a Bluetooth® platform configuration structure that is provided in the BSP. By default, the file can be found in the BSP inside the application at:

<application_root>/libs/TARGET_<bsp_name>/bluetooth/cybsp_bt_config.c

This file contains the `cybsp_bt_platform_cfg` structure that you use in the call to `chybt_platform_config_init` that you learned about in an earlier chapter. The contents of that file look like this:

```
#if defined(COMPONENT_WICED_BLE) || defined(COMPONENT_WICED_DUALMODE)

#include "cybsp_bt_config.h"
#include "wiced_bt_dev.h"

const cybt_platform_config_t cybsp_bt_platform_cfg =
{
    .hci_config                             =
    {
        .hci_transport                      = CYBT_HCI_UART,

        .hci                                =
        {
            .hci_uart                       =
            {
                .uart_tx_pin                = CYBSP_BT_UART_TX,
                .uart_rx_pin                = CYBSP_BT_UART_RX,
                .uart_rts_pin               = CYBSP_BT_UART_RTS,
                .uart_cts_pin               = CYBSP_BT_UART_CTS,

                .baud_rate_for_fw_download  = CYBSP_BT_PLATFORM_CFG_BAUD_DOWNLOAD,
                .baud_rate_for_feature      = CYBSP_BT_PLATFORM_CFG_BAUD_FEATURE,

                .data_bits                  = CYBSP_BT_PLATFORM_CFG_BITS_DATA,
                .stop_bits                  = CYBSP_BT_PLATFORM_CFG_BITS_STOP,
                .parity                     = CYHAL_UART_PARITY_NONE,
                .flow_control               = true
            }
        }
    },

    .controller_config                      =
    {
        .bt_power_pin                       = CYBSP_BT_POWER,
        .sleep_mode                         =
        {
            .sleep_mode_enabled             = CYCFG_BT_LP_ENABLED,
            .device_wakeup_pin              = CYCFG_BT_DEV_WAKE_GPIO,
            .host_wakeup_pin                = CYCFG_BT_HOST_WAKE_GPIO,
            .device_wake_polarity           = CYCFG_BT_DEV_WAKE_POLARITY,
            .host_wake_polarity             = CYCFG_BT_HOST_WAKE_IRQ_EVENT
        }
    },

    .task_mem_pool_size                     = CYBSP_BT_PLATFORM_CFG_MEM_POOL_BYTES

};
```

The first thing to notice here is that the code is only included if either BLE or Dual-Mode is enabled for the Bluetooth® device. This structure defines the pins and settings for the UART interface and the low power interface, as well as the size of the pool of memory for the Bluetooth® task.

8.3 Advertising

As you will see in the exercises, advertising can take a significant amount of power. If your device is expected to advertise frequently, reducing the advertising duty cycle as much as possible will reduce power consumption.

8.4 Exercises

The examples that you have been working with up to this point already had some low power features enabled. Namely, low power modes were enabled for both the MCU and connectivity devices. Other changes to reduce overall system power consumption include changing to lower power regulator modes, disabling unused features such as CAPSENSE™ or debug mode, or slowing down or even disabling some clocks. These are features that have NOT been done up to this point.

For these exercises, you will start by measuring the current consumed by the PSoC and the connectivity device using the basic BLE exercise. Then you will turn off the low power modes on each device to see how much the current consumption increases.

Then, you will create an application that has low power modes enabled and adds on additional power savings by disabling unused resources and slowing clocks.

Note: *When using the Device Configurator, remember that the files it uses are typically part of the BSP so be aware that any edits you make may result in a dirty git repo for the BSP if you are using a standard BSP. Refer to the ModusToolbox™ Software Training Level 1 - Getting Started class for details on how to create a custom BSP or override the configuration from the BSP within a single application.*

Exercise 1: Sleep Mode Impact on Power

In this exercise, you will measure the current consumed by the MCU and connectivity device when the basic BLE exercise is running. You will then disable sleep modes on the two devices to see what impact that has on the current consumption.

Hardware setup

The ModusToolbox™ level 2 PSoC™ class has a low power chapter that explains the hardware setup to measure current consumption on the CY8CKIT-062S2-43012. Refer to that chapter for additional details if needed.

- [ ] 1. Disconnect power from the kit.
- [ ] 2. Remove the shunt on the connectivity device current measurement jumper (VBAT)(J8) and connect an ammeter across the two terminals.
- [ ] 3. Remove the shunt on the MCU current measurement jumper (P6.VDD)(J15) and connect an ammeter across the two terminals.

Note: *If you only have one ammeter available, you can move it between the two jumpers. Just remember to disconnect the board before changing the connections and put the shunt back on the jumper not connected to the ammeter.*

4. Remove the shunt on the potentiometer power pin (VDD.POT)(J25) so that its current isn't included in the MCU current measurement.

5. Reconnect power to the kit.

Note: *Turn on the ammeters before powering the kit since some ammeters are high impedance when turned off.*

Note: *Do not disconnect or turn off the ammeters while the kit is powered.*

Application creation - sleep modes enabled

1. Create a new ModusToolbox™ application for the CY8CKIT-062S2-43012 BSP.

   Use the Import functionality in project creator to start from the completed application for exercise 1 from chapter 3. If you did not complete that exercise, the solution can be found in *Projects/key_ch03_ex01_ble*. Name the new application *ch08_ex01_no_low_power*.

2. Open the Bluetooth® configurator and change the device name to **<inits>_no_low_power**.

3. Save changes and close the configurator.

Testing - sleep modes enabled

1. Program the application to your kit.

2. Write down current values that you see for both device power supplies for advertising off, low duty cycle advertising, high duty cycle advertising, and connected. If your ammeter is capable of measuring the average current, use that value. If not, just record the max value that you see.

Note: *A table is provided below to record your results.*

Results - sleep modes enabled

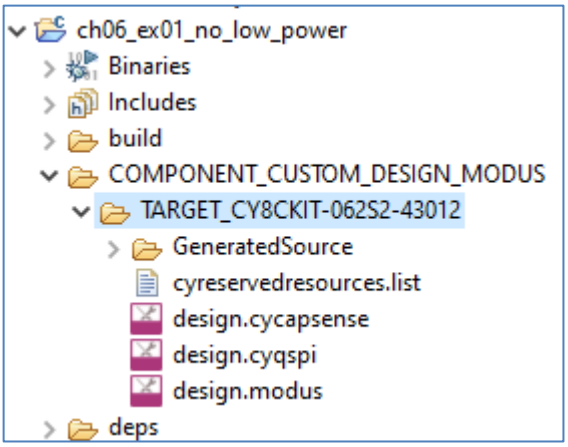
Current (uA)	PSoC 6	CYW43012
Advertising Stopped		
Low Duty Cycle Advertising		
High Duty Cycle Advertising		
Connected		

Now let's see how much the sleep modes are helping to reduce power consumption. First, we will copy the device configuration from the BSP into the application so that changes will not affect the BSP.

Application creation – sleep modes disabled

1. Create a new directory in the project's root directory called *COMPONENT_CUSTOM_DESIGN_MODUS* and a sub-directory under that called *TARGET_CY8CKIT-062S2-43012*.

2. Copy all files from *libs/TARGET_CY8CKIT-062S2-43012/COMPONENT_BSP_DESIGN_MODUS/* to the new directory that you created.
   The hierarchy should look like this when you finish:



3. Edit the application's Makefile as follows and save it when you are done:

   a. Add `CUSTOM_DESIGN_MODUS` to the `COMPONENTS` variable:
   `COMPONENTS=FREERTOS WICED_BLE CUSTOM_DESIGN_MODUS`

   b. Add `BSP_DESIGN_MODUS` to the `DISABLE_COMPONENTS` variable:
   `DISABLE_COMPONENTS=BSP_DESIGN_MODUS`

4. If you are using the Eclipse IDE for ModusToolbox™, click on the **Refresh Quick Panel** link so that the correct device configurator files will open when you edit them.

5. Open the device configurator. Check in the banner that the correct file is open – it should be the one in *COMPONENT_CUSTOM_DESIGN_MODUS.*

6. On the tab for the MCU device, change the **Power > RTOS > System Idle Power Mode** to **Active.**

   This will prevent the MCU from entering sleep or deep sleep mode.

7. On the tab for the connectivity device, uncheck the box under **Power > BT**.

   This will prevent the connectivity device from entering sleep mode.

8. Save edits and close the configurator.

Testing – sleep modes disabled

☐ 1. Program the application to your kit.

Write down the current values that you see for both device power supplies when advertising, when advertising times out, and when connected. If your ammeter is capable of measuring the average current, record that as well.

☐ 2. Compare the values to the ones you recorded for the baseline.

You should see MUCH higher current consumption for both devices when sleep modes are disabled.

Results – sleep modes disabled

Current (uA)	PSoC 6	CYW43012
Advertising Stopped		
Low Duty Cycle Advertising		
High Duty Cycle Advertising		
Connected		

Exercise 2: Enable low power, use lower power regulator modes, disable unused resources, and slow or stop clocks

In this exercise, you will disable unused resources and slow down the MCU clocks to further reduce current consumption.

1. Create a new ModusToolbox™ application for the CY8CKIT-062S2-43012 BSP.

   Use the Import functionality in project creator to start from the completed application for exercise 1 from this chapter. If you did not complete that exercise, the solution can be found in *Projects/key_ch08_ex01_no_low_power*.

   Name the new application *ch08_ex02_low_power*.

2. Open the device configurator. Verify in the banner that the custom configuration for the application is open, not the one from the BSP.

3. Re-enable sleep modes for both the MCU and connectivity devices.

   a. On the tab for the MCU device, change **the Power > General > System Idle Power Mode** to **System Deep Sleep.**

   b. On the tab for the connectivity device, check the box under **Power > BT**.

4. Change regulator modes.

Note: *If you need more details on how to make the remaining changes, refer to the low power chapter of the ModusToolbox™ PSoC™ level 2 training class.*

Note: *Changing the regulator modes will cause errors to appear in the notification area. Don't worry – these will be fixed in the next step.*

   a. Change the **System Active Power** mode to **ULP**.

   b. Change the **Core Regulator** to **Minimum Current Buck**.

5. Slow clock frequencies and disable unused system clocks.

Note: *Turning off some of the clocks will cause errors to appear in the notification area. Don't worry – they will be fixed in the next step.*

   a. Change the FLL frequency to 24 MHz.

   b. Disable the following clocks under **System > System Clocks**:
   PLL 0, CLK_SLOW , ILO, CLK_ALT_SYS_TICK, CLK_BAK, CLK_TIMER

   c. Under **Peripheral-Clocks**, disable 8-bit Divider 0

6. Disable unused resources.

   a. Under **Peripherals > System**, turn off the CAPSENSE™ block (**CSD**).

   b. Under System, uncheck the box for **Debug** and disable the debug pins.

   c. Under **Pins**, disable the CAPSENSE™ and Debug pins. The only pins still enabled should be the two WCO pins.

Note: *Use the **Filter** button to show only the enabled pins.*

7. Save changes and close the configurator.
8. In the Makefile, change the value of the `CONFIG` variable from `Debug` to `Release` to change the compiler's optimization settings. Save your changes.

```
CONFIG=Release
```

9. If you are using the Eclipse IDE for ModusToolbox™, in the Quick Panel under the Launches section, click *Generate Launches ch08_ex06_low_power*.

Note: *This is necessary so that the program link will use the correct files (from the Release build directory) to program the kit.*

10. Open the Bluetooth® configurator
11. Change the **Device name** to **<inits>_low_power**.
12. Set the **High duty advertising interval minimum** and **High duty advertising interval maximum** both to 250.
13. Program the application to your kit. Reset the kit once programming is done to ensure a clean boot up sequence.

    Write down the current values that you see for both device power supplies when advertising, when advertising times out, and when connected. If your ammeter is capable of measuring the average current, record that as well.
14. Compare the values to the ones you recorded in exercise 1.

    You should see lower current consumption for the MCU compared to the prior results.

Results

Current (uA)	PSoC 6	CYW43012
Advertising Stopped		
Low Duty Cycle Advertising		
High Duty Cycle Advertising		
Connected		

Note: *The file low_power_measurement.docx in the solution project directory has example values that were measured for these exercises for comparison.*

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

Published by

Infineon Technologies AG

81726 Munich, Germany

IMPORTANT NOTICE

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics (“Beschaffenheitsgarantie”) .

With respect to any examples, hints or any typical values stated herein and/or any information regarding the application of the product, 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.

In addition, any information given in this document is subject to customer’s compliance with its obligations stated in this document and any applicable legal requirements, norms and standards concerning customer’s products and any use of the product of Infineon Technologies in customer’s applications.

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.