Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

CE217636 - BLE Environmental Sensing Profile with PSoC 6 MCU with BLE Connectivity

Objective

This example demonstrates the BLE Environmental Sensing Profile application workflow.

Overview

This example project demonstrates the Environmental Sensing Profile operation of the BLE PSoC Creator™ Component. The Environmental Sensor uses the Environmental Sensing Profile with one instance of Environmental Sensing and Device Information Services to simulate measuring the wind speed. The Environmental Sensor operates with other devices that implement the Environmental Collector Profile. The device switches to Deep Sleep mode between BLE connection intervals. The BLE Component supports PSoC 6 BLE.

Requirements

Tool: PSoC Creator™ 4.2

Programming Language: C (ARM® GCC 5.4-2016-q2-update)

Associated Parts: All PSoC 6 BLE parts

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

Hardware Setup

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

1. Connect the BLE Pioneer Kit to the computer's USB port.
2. Connect the BLE Dongle to one of the USB ports on the computer.

LED Behavior

If the VDDD voltage is set to less than 2.7 V in the DWR settings **System** tab, only the red LED is used. The red LED blinks to indicate that the device is advertising. The red LED is OFF when a device is connected to a peer device. When the device is in Hibernate mode, the red LED stays ON.

LED behavior for VDDD Voltage > 2.7 volts is described in **Operation** section.

Software Setup

BLE Host Emulation Tool

This example requires the CySmart application. Download and install either the CySmart Host Emulation Tool PC application or the CySmart app for iOS or Android. You can test 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

Terminal Tool

This example uses a terminal window. You must have terminal software, such as Tera Term, or PuTTy.

Operation

Simulation of Measuring Wind Speed and Humidity

This example project simulates the data measured from two wind speed sensors (True Wind Speed Speed#1 and True Wind Speed Speed#2) and humidity sensor. Each sensor is configured to provide a new measurement with respect to its Update Interval set in the ES Measurement Descriptor. The Update Intervals for True Wind Speed Speed#1and #2 are configured for 15 and 5 seconds respectively. The Update Interval for Humidity is configured for 10 seconds. Each sensor also has its Measurement Period: 60 seconds – for True Wind Speed Speed#1; 90 seconds – for True Wind Speed Speed#2; 40 seconds – for Humidity. For more details on the Measurement Period and Update Interval, refer to the Environmental Sensing Service specification.

The device is configured to send notifications to a remote Client based on the trigger conditions captured in up to three ES Trigger Settings Descriptors. Depending on the configuration in the ES Configuration Descriptor, the conditions in the ES Trigger Settings Descriptors can be ORed or ANDed. The User Characteristic Descriptor is used for assigning a human-readable name of the Characteristic. The ES Trigger Settings, and the ES Configuration and User Characteristic Descriptor Descriptors are writable and can be set by a remote Client.

The example project allows configuring a simulation via the Customizer's GUI. The Measuring Period and Update Interval can be set in the ES Measurement Descriptor. The notification conditions can be configured in the ES Trigger Settings Descriptor. Each characteristic can be assigned with a default name through the User Characteristic Descriptor. The Valid Range Descriptor can be used to define the allowed ranges for the characteristic. In the current example project, the ranges are set to the maximum possible values.

The first wind sensor simulates an increase in the wind speed by 1.2 m/s every 15 seconds until it reaches the maximum of 80 m/s. Then the wind speed drops to the minimum of 10 m/s, and then again it is increased by 1.2 m/s every 15 seconds. The second wind sensor simulates an increase in the wind speed by 0.2 m/s every 5 seconds until it reaches the maximum of ~90 m/s. However, notification of this parameter is every 20 seconds because the ES Trigger Settings Descriptor is set to "No less than the specified time between transmissions" in this example project and the specified time is set to 20 seconds.

The humidity sensor simulates an increase in the humidity by 1.40% every 10 seconds until it reaches the maximum of 99.00%. Then the humidity drops to the minimum of 2%, and then again it is increased by 1.40% every 10 seconds. However, a value between 20.00% and 40.00% is not notified because the ES Trigger Settings Descriptor is set to "While less than the specified value" (specified value is set to 20.00), ES Trigger Settings Descriptor 2 is set to "While greater than the specified value" (specified value is set to 40.00) and the ES Configuration is set to "Boolean OR" for humidity in this example project. After that, the speed is not updated any more holding the maximum wind speed.

Refer to the Environmental Sensing Service Specification for more details.

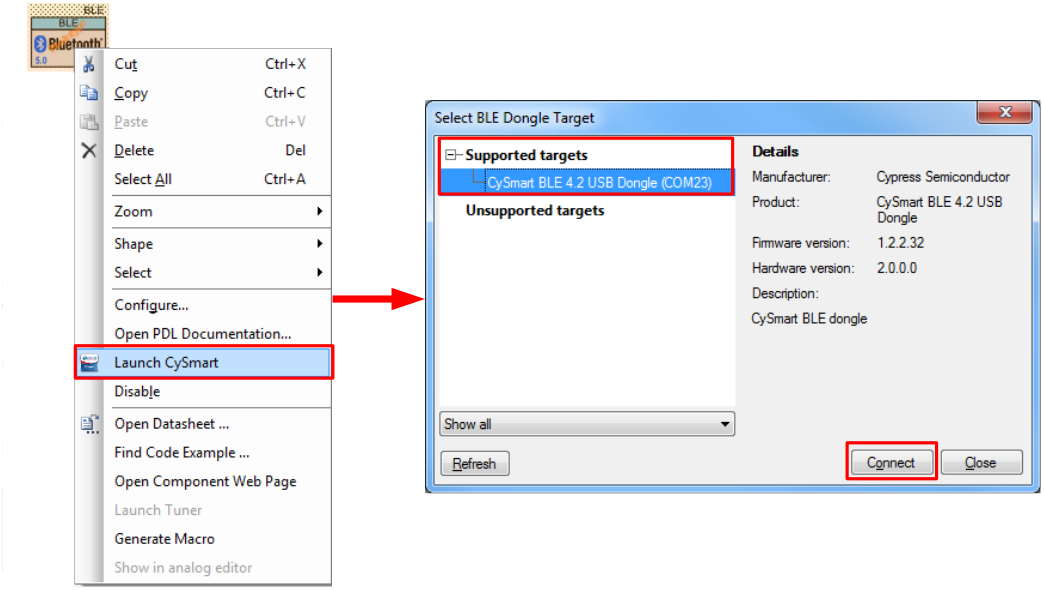
You can use the CySmart app on a Windows PC, Android, or iOS BLE-compatible device as a Client for connection to the Weight Scale.

Operation Steps

1. Plug the CY8CKIT-062-BLE kit board into your computer's USB port.
2. Open a terminal window and perform following configuration: Baud rate – 115200, Parity – None, Stop bits – 1, Flow control – XON/XOFF. These settings must match the configuration of the PSoC Creator UART Component in the project.
3. 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.
4. Observe the green LED blinks while the device is advertising, and the output in the terminal window.
5. Do the following to test example, using the CySmart Host Emulation Tool application as Environmental Sensing Service Client:
    a. Connect the BLE Dongle to your Windows PC. Wait for the driver installation to complete, if necessary.
    b. Launch the CySmart Host Emulation Tool by right-clicking on the BLE Component and selecting **Launch CySmart**. Alternatively, you can launch the tool by navigating to **Start** > **Programs** > **Cypress** and clicking on **CySmart**.

c. CySmart automatically detects the BLE dongle 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 1.

Figure 1. CySmart BLE Dongle Selection

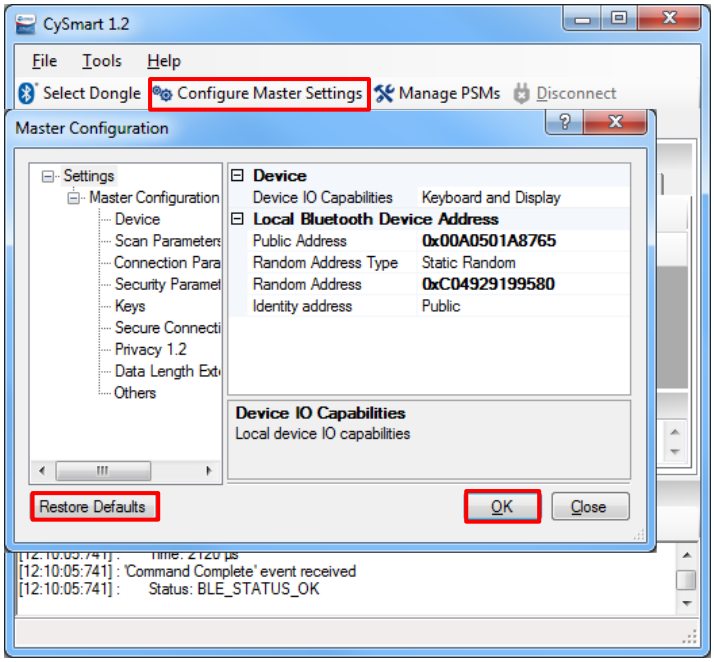


Note: If the dongle firmware is outdated, you will be alerted with an appropriate message. You must upgrade the firmware before you can complete this step. Follow the instructions in the window to update the dongle firmware.

■

d. Select **Configure Master Settings** and then click **Restore Defaults**, as Figure 2 shows. Then click **OK**.

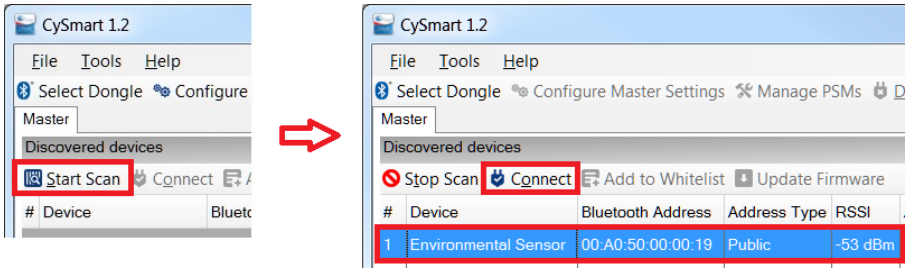
Figure 2. CySmart Master Settings Configuration



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

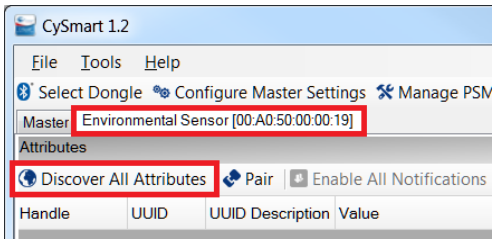
f. On the CySmart Host Emulation Tool, click **Start Scan**. Your device name (configured as **Environmental Sensor**) should appear in the Discovered devices list, as Figure 3 shows. Select the device and click **Connect** to establish a BLE connection between the CySmart Host Emulation Tool and your device.

Figure 3. CySmart Device Discovery and Connection



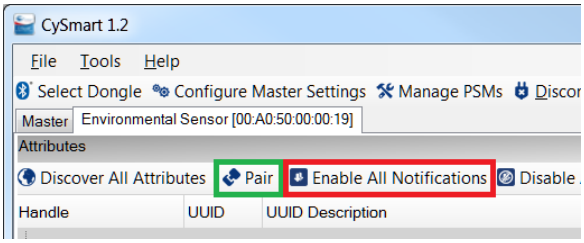
g. Once connected, switch to the '**Environmental Sensor**' device tab and '**Discover all Attributes**' on your design from the CySmart Host Emulation Tool, as shown in Figure 4.

Figure 4. CySmart Attribute Discovery



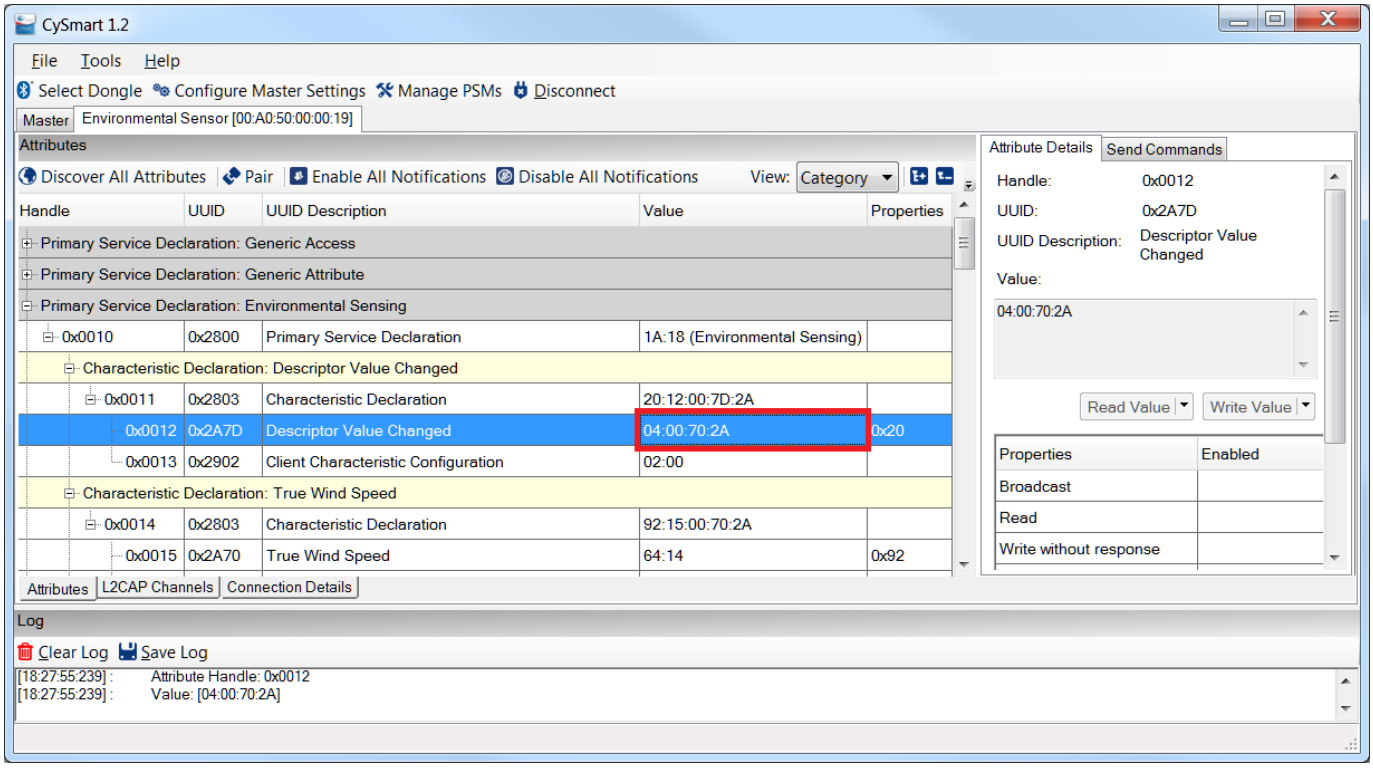
h. Click **Pair** after discovery finishes, then **Enable All Notifications** in the CySmart app as shown in Figure 5.

Figure 5. CySmart Pair and Enable All Notification



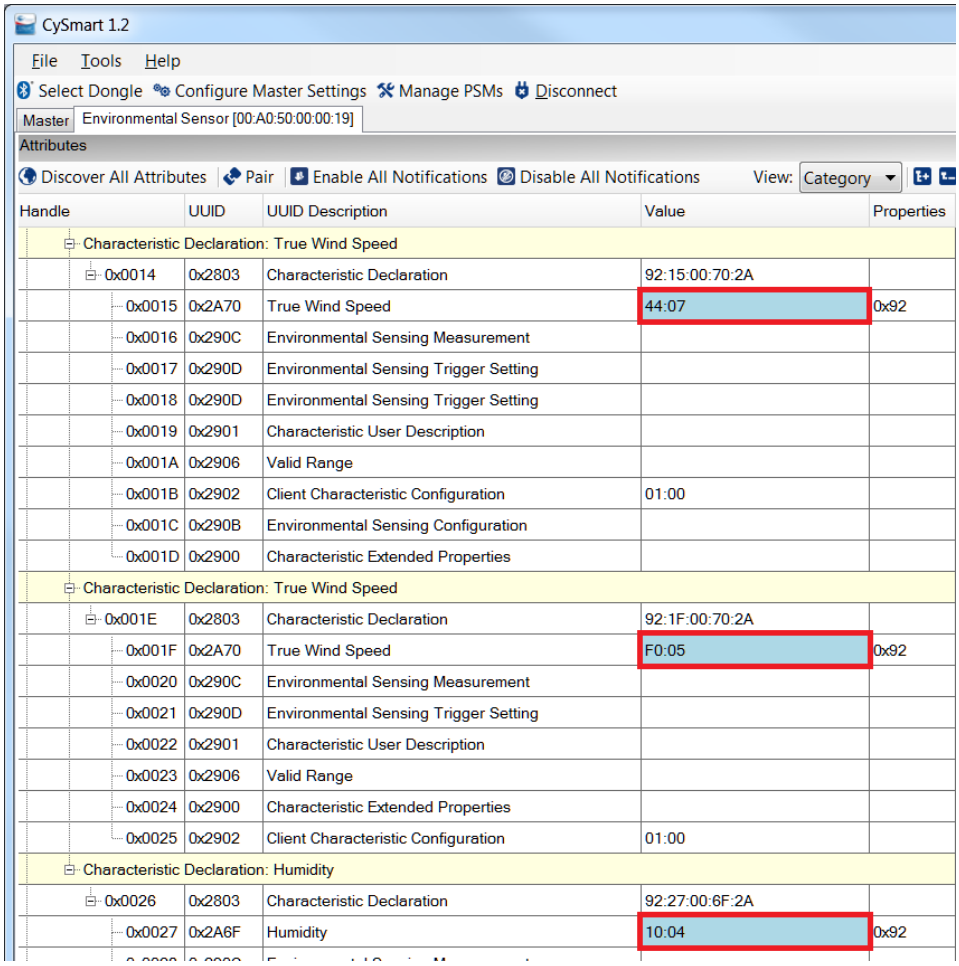
i. Press the **SW2** button on the on the CY8CKIT-062 kit and observe the indication about ES Configuration Descriptor change:

Figure 6. CySmart Windows App: Descriptor Value Changed Characteristic Indication



j. Wait for at least 90 seconds (simulation of the measurement period) and observe that notifications for the True Wind Speed and Humidity Characteristics are received:

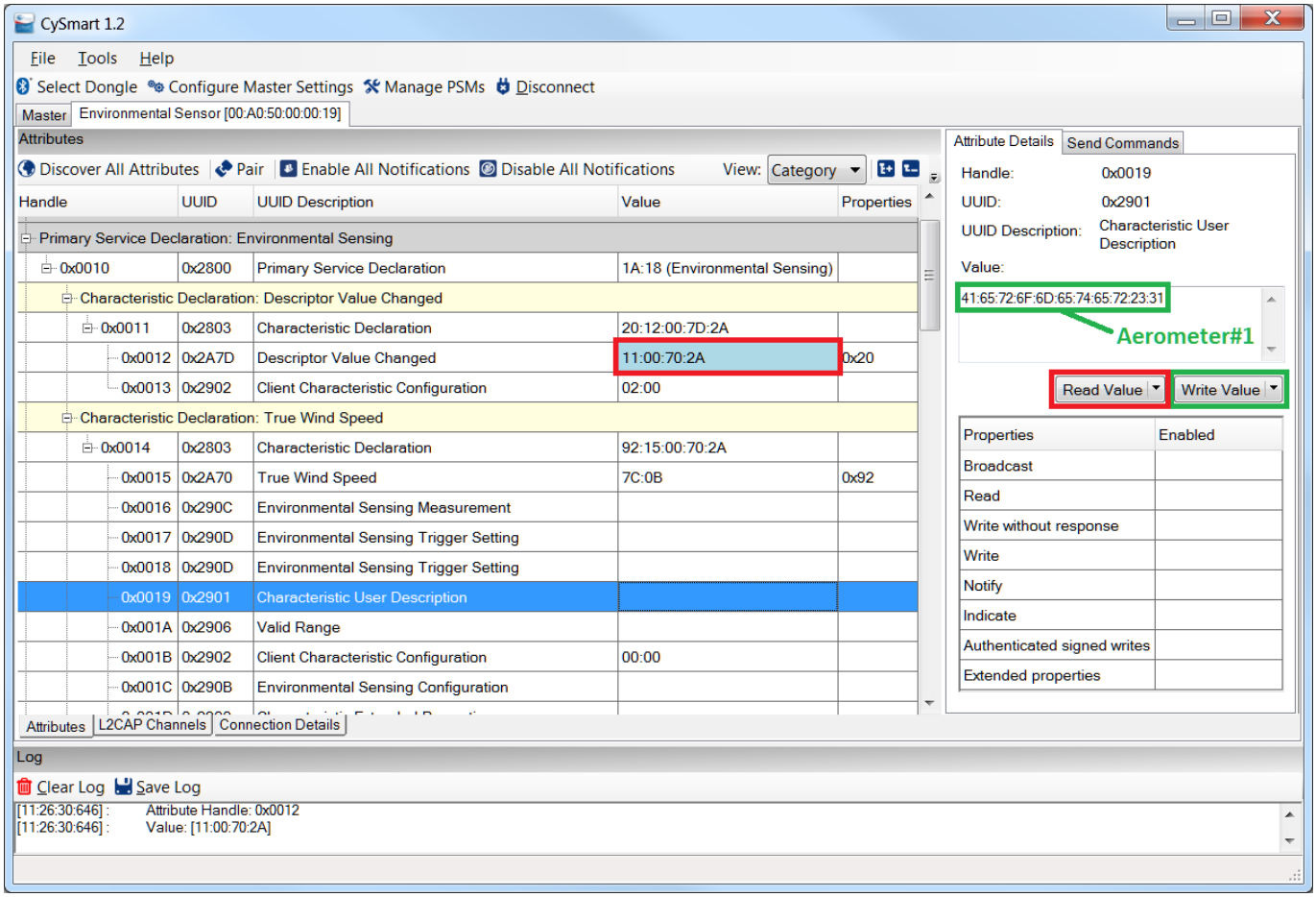
Figure 7. CySmart Windows App: True Wind Speed and Humidity Characteristics Notification



Handle	UUID	UUID Description	Value	Properties
Characteristic Declaration: True Wind Speed				
0x0014	0x2803	Characteristic Declaration	92:15:00:70:2A	
0x0015	0x2A70	True Wind Speed	44:07	0x92
0x0016	0x290C	Environmental Sensing Measurement		
0x0017	0x290D	Environmental Sensing Trigger Setting		
0x0018	0x290D	Environmental Sensing Trigger Setting		
0x0019	0x2901	Characteristic User Description		
0x001A	0x2906	Valid Range		
0x001B	0x2902	Client Characteristic Configuration	01:00	
0x001C	0x290B	Environmental Sensing Configuration		
0x001D	0x2900	Characteristic Extended Properties		
Characteristic Declaration: True Wind Speed				
0x001E	0x2803	Characteristic Declaration	92:1F:00:70:2A	
0x001F	0x2A70	True Wind Speed	F0:05	0x92
0x0020	0x290C	Environmental Sensing Measurement		
0x0021	0x290D	Environmental Sensing Trigger Setting		
0x0022	0x2901	Characteristic User Description		
0x0023	0x2906	Valid Range		
0x0024	0x2900	Characteristic Extended Properties		
0x0025	0x2902	Client Characteristic Configuration	01:00	
Characteristic Declaration: Humidity				
0x0026	0x2803	Characteristic Declaration	92:27:00:6F:2A	
0x0027	0x2A6F	Humidity	10:04	0x92

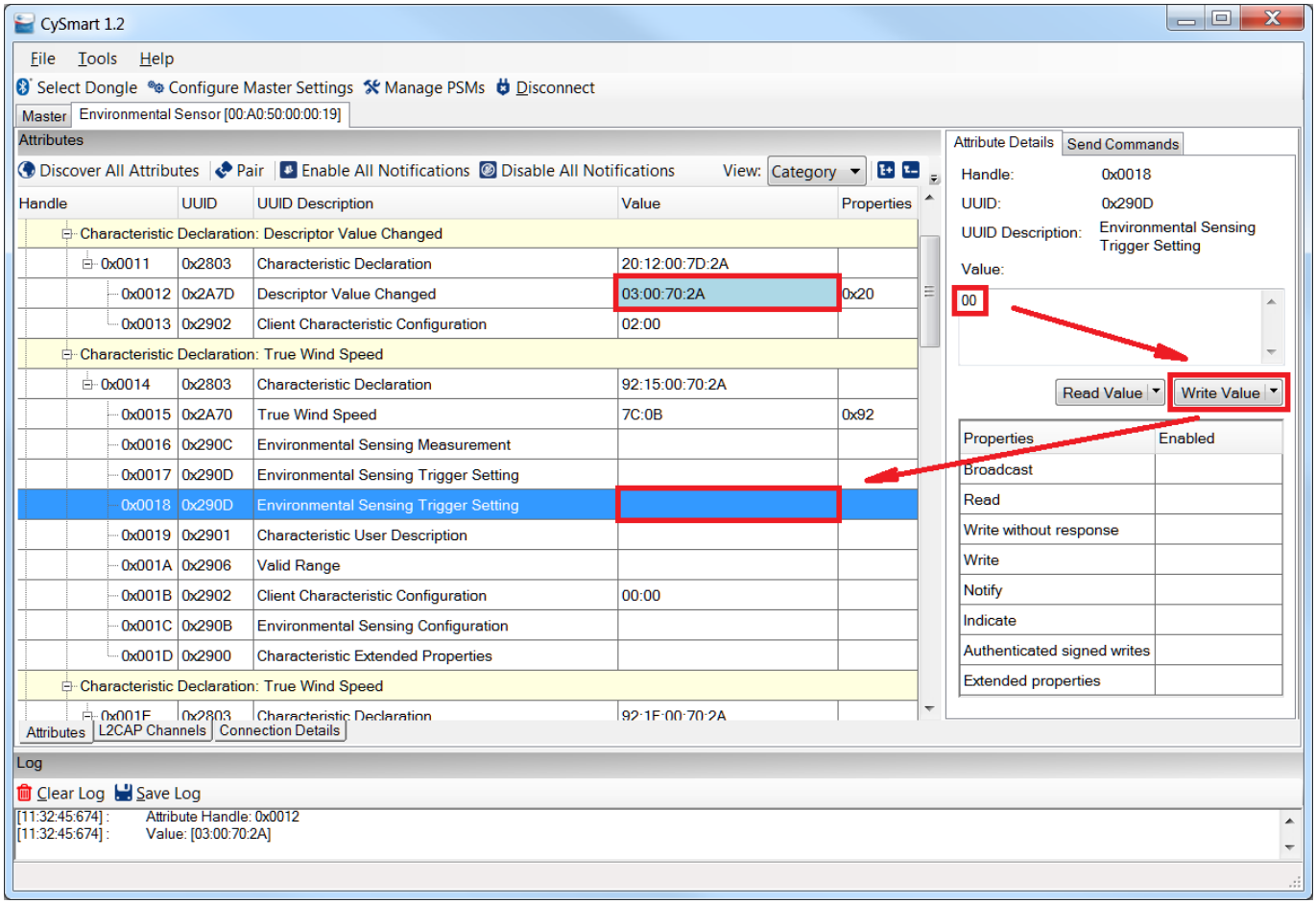
k. Write a human-readable name for a Characteristic to the Characteristic User Description Descriptor (UUID 0x2901), press the **Write Value** button, and then press **Read Value** and observe the Descriptor Value Changed Characteristic indication with regard to that. Before writing, convert the descriptor value to the ASCII numbers first. In Figure 8, the name “Aerometer#1” – 41:65:72:6F:6D:65:74:65:72:23:31 (ASCII) is used.

Figure 8. CySmart Windows App: Writing to Characteristic User Description Descriptor



l. Write the value 0x00 to the second ES Trigger Settings Descriptor (UUID 0x290D), press the **Write Value** button, and observe the Descriptor Value Changed Characteristic indication with regard to that:

Figure 9. CySmart Windows App: Writing to ES Trigger Descriptor



6. The CySmart mobile app (Android/iOS) does not have Environmental Sensing Profile implementation, but still can be used in GATT Data Base mode for test this example. You can repeat test flow for CySmart mobile app in step 5. Refer to Android and iOS CySmart User Guide.
7. Use the UART debug port to view verbose messages:
   a. The code example ships with the UART debug port enabled. To disable it, set the macro DEBUG_UART_ ENABLED in *common.h* to DISABLED and rebuild the code.
   b. The output of the debug serial port looks like the sample below.

BLE Environmental Sensing Profile Example

```
* The initialized Characteristic - True Wind Speed instance #1
* Value of imitated parameter          - 1500
* Maximum value of imitated parameter  - 8000
* Minimum value of imitated parameter  - 1000
* Step of imitated parameter changing   - 120
* Value of ES Configuration descriptor  - AND
* Notification timeout value            - 10
* Value condition, Comparison value #0  - 1, 1048592
* Value condition, Comparison value #1  - 3, 1048576
* Value condition, Comparison value #2  - 0, 0
```

```
* Measurement period in seconds          - 60
* Update Interval in seconds             - 15

* The initialized Characteristic - True Wind Speed instance #2
* Value of imitated parameter            - 1500
* Maximum value of imitated parameter    - 9000
* Minimum value of imitated parameter    - 700
* Step of imitated parameter changing    - 20
* Value of ES Configuration descriptor   - AND
* Notification timeout value             - 20
* Value condition, Comparison value #0   - 2, 20
* Value condition, Comparison value #1   - 0, 0
* Value condition, Comparison value #2   - 0, 0
* Measurement period in seconds          - 90
* Update Interval in seconds             - 5

* The initialized Characteristic - Humidity instance #1
* Value of imitated parameter            - 200
* Maximum value of imitated parameter    - 9900
* Minimum value of imitated parameter    - 200
* Step of imitated parameter changing    - 140
* Value of ES Configuration descriptor   - AND
* Notification timeout value             - 10
* Value condition, Comparison value #0   - 4, 2000
* Value condition, Comparison value #1   - 6, 4000
* Value condition, Comparison value #2   - 0, 0
* Measurement period in seconds          - 40
* Update Interval in seconds             - 10

CY_BLE_EVT_STACK_ON, StartAdvertisement
CY_BLE_EVT_SET_DEVICE_ADDR_COMPLETE
CY_BLE_EVT_LE_SET_EVENT_MASK_COMPLETE
CY_BLE_EVT_GET_DEVICE_ADDR_COMPLETE: 00a050000019
CY_BLE_EVT_SET_TX_PWR_COMPLETE
CY_BLE_EVT_SET_TX_PWR_COMPLETE
CY_BLE_EVT_GAPP_ADVERTISEMENT_START_STOP, state: 2
CY_BLE_EVT_GAP_KEYS_GEN_COMPLETE
CY_BLE_EVT_GATT_CONNECT_IND: 0, 8
CY_BLE_EVT_GAP_DEVICE_CONNECTED: connIntv = 7 ms
CY_BLE_EVT_GATTS_XCNHG_MTU_REQ 0, 8, final mtu= 23
CY_BLE_EVT_GATTS_READ_CHAR_VAL_ACCESS_REQ: handle: 3
CY_BLE_EVT_GAP_AUTH_REQ: bdHandle=8, security=3, bonding=1, ekeySize=10, err=0
CY_BLE_EVT_GAP_SMP_NEGOTIATED_AUTH_INFO: bdHandle=8, security=1, bonding=1, ekeySize=10, err=0
CY_BLE_EVT_STACK_BUSY_STATUS: 1
CY_BLE_EVT_GAP_ENCRYPT_CHANGE: 0
CY_BLE_EVT_STACK_BUSY_STATUS: 0
CY_BLE_EVT_GAP_KEYINFO_EXCHNGE_CMPLT
CY_BLE_EVT_GAP_AUTH_COMPLETE: bdHandle=8, security=1, bonding=1, ekeySize=10, err=0
CY_BLE_EVT_PENDING_FLASH_WRITE
Store bonding data, status: 140001, pending: 1
Store bonding data, status: 140001, pending: 1
Store bonding data, status: 140001, pending: 1
Store bonding data, status: 140001, pending: 1
Store bonding data, status: 140001, pending: 1
Store bonding data, status: 140001, pending: 1
Store bonding data, status: 140001, pending: 1
Store bonding data, status: 140001, pending: 1
Store bonding data, status: 0, pending: 0
CY_BLE_EVT_GATTS_INDICATION_ENABLED Store bonding data, status: 0, pending: 0
CY_BLE_EVT_GATTS_READ_CHAR_VAL_ACCESS_REQ: handle: f
```

```
CY_BLE_EVT_ESSS_NOTIFICATION_ENABLED
Char instance: 1
Store bonding data, status: 0, pending: 0
CY_BLE_EVT_GATTS_READ_CHAR_VAL_ACCESS_REQ: handle: 2c
CY_BLE_EVT_ESSS_NOTIFICATION_ENABLED
Char instance: 2
Store bonding data, status: 0, pending: 0
CY_BLE_EVT_GATTS_READ_CHAR_VAL_ACCESS_REQ: handle: 25
CY_BLE_EVT_ESSS_NOTIFICATION_ENABLED
Char instance: 1
Store bonding data, status: 0, pending: 0
CY_BLE_EVT_GATTS_READ_CHAR_VAL_ACCESS_REQ: handle: 1b
CY_BLE_EVT_ESSS_INDICATION_ENABLED

Store bonding data, status: 0, pending: 0
CY_BLE_EVT_GATTS_READ_CHAR_VAL_ACCESS_REQ: handle: 13
Measurement Period for Humidity sensor#1 (40 s) has elapsed.
Update Interval for Humidity sensor#1 (10 s) has elapsed.
Update Interval for Humidity sensor#1 (10 s) has elapsed.
Measurement Period for True Wind Speed sensor#1 (60 s) has elapsed.
Update Interval for True Wind Speed sensor#1 (15 s) has elapsed.
Update Interval for Humidity sensor#1 (10 s) has elapsed.
Notification for True Wind Speed #1 was sent successfully. Notified value is: 16.20 m/s.
Update Interval for Humidity sensor#1 (10 s) has elapsed.
Update Interval for True Wind Speed sensor#1 (15 s) has elapsed.
Notification for True Wind Speed #1 was sent successfully. Notified value is: 17.40 m/s.
Update Interval for Humidity sensor#1 (10 s) has elapsed.
Update Interval for True Wind Speed sensor#1 (15 s) has elapsed.
Measurement Period for True Wind Speed sensor#2 (90 s) has elapsed.
Update Interval for True Wind Speed sensor#2 (5 s) has elapsed.
Update Interval for Humidity sensor#1 (10 s) has elapsed.
Notification for True Wind Speed #1 was sent successfully. Notified value is: 18.60 m/s.
Notification for True Wind Speed #2 was sent successfully. Notified value is: 15.20 m/s.
Notification for Humidity #1 was sent successfully. Notified value is: 10.40 %%.
Indication for ES Configuration Descriptor was sent successfully.
Indicated value is: 2A 70 00 04
CY_BLE_EVT_ESSS_INDICATION_CONFIRMATION
Update Interval for True Wind Speed sensor#2 (5 s) has elapsed.
```

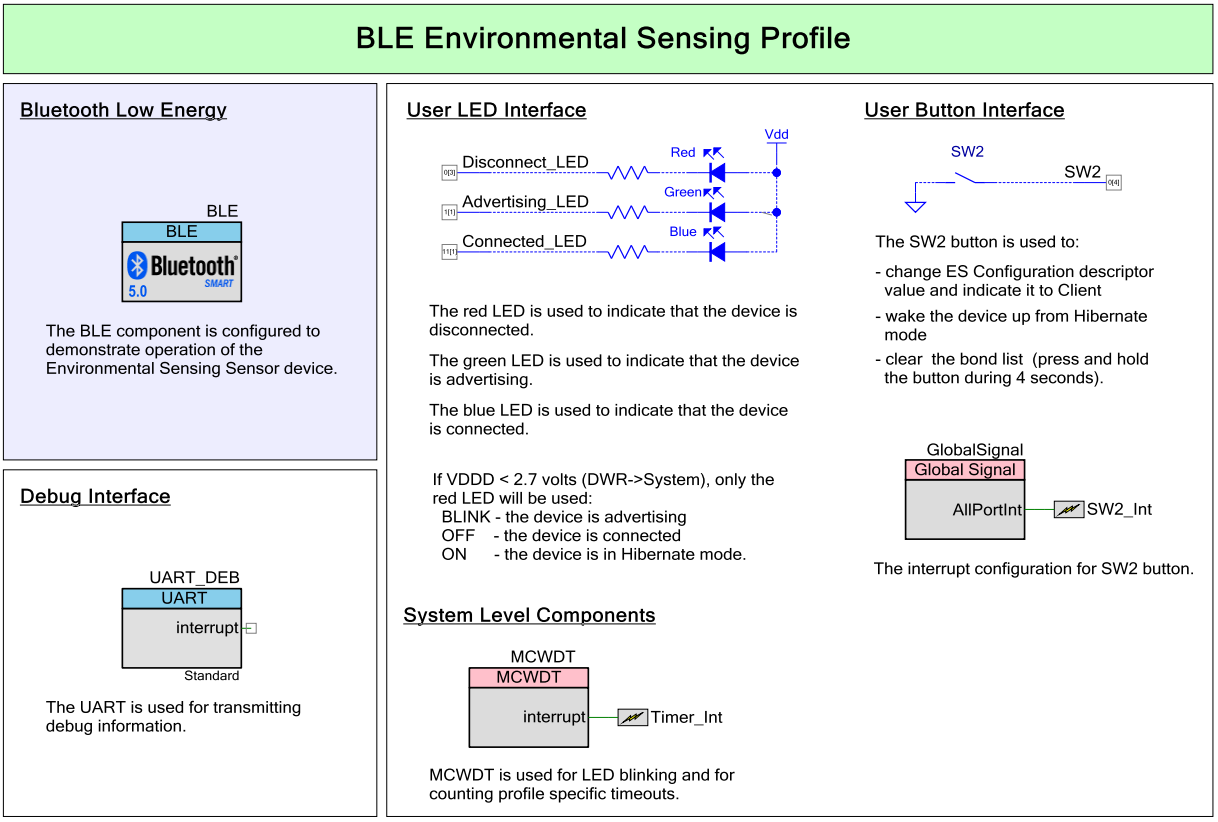
The details about the Environmental Sensing Service characteristic data structures are in the ESS Specification.

If you have problems with the usage of the CySmart app, refer to the CySmart User Guide.

Design and Implementation

This example project demonstrates the Environmental Sensing Profile operation of the BLE PSoC Creator Component. The Environmental Sensor uses the Environmental Sensing Profile with one instance of Environmental Sensing and Device Information Services to simulate measuring the wind speed. The Environmental Sensor operates with other devices that implement the Environmental Collector Profile. Figure 10 shows the top design schematic.

Figure 10. BLE Environmental Sensing Profile Code Example Schematic



The project demonstrates the core functionality of the BLE Component configured as an Environmental Sensor.

After a startup, the device performs initialization of the BLE Component. In this project, several callback functions are used for the BLE operation. One callback function (AppCallBack()) is required for receiving generic events from the BLE stack, EssCallBack() is required for receiving events from the Environmental Service. The CY_BLE_EVT_STACK_ON event indicates a successful initialization of the BLE stack. After this event is received, the Component starts fast advertising with the packet structure as configured in BLE Component Customizer. After the 30-second advertising period expires, the Component switches to slow advertisement parameters. On an advertisement event timeout, the device goes to the Hibernate low-power mode and waits for a **SW2** button press to wake up the device again.

You can connect to the Environmental Sensor device with a BLE 4.0 or BLE 4.1-compatible device configured in the GAP Central role and capable of discovering the Environmental Sensing Service. To connect to an Environmental Sensor device, send a connection request to the device when the device is advertising. The green LED blinks while the device is advertising. If the Client is connected to the Environmental Sensor, the blue LED is turned ON.

While connected to the Client and between the connection intervals, the device is put into Deep Sleep mode.

Pin assignments

Pin assignments and connections required on the development board for supported kits are in Table 1.

Table 1. Pin Assignment

Pin Name	Development Kit	Comment
	CY8CKIT-062	
\UART_DEB:rx\	P5[0]	
\UART_DEB:tx\	P5[1]	
\UART_DEB:rts\	P5[2]	
\UART_DEB:cts\	P5[3]	
Advertising_LED	P1[1]	The green color of the RGB LED
Disconnect_LED	P0[3]	The red color of the RGB LED
Connected_LED	P11[1]	The blue color of the RGB LED
SW2	P0[4]	

Components and Settings

Table 2 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 2. PSoC Creator Components

Component	Instance Name	Purpose	Non-default Settings
Bluetooth Low Energy (BLE)	BLE	The BLE component is configured to demonstrate operation of the Environmental Sensing Sensor device.	Refer to Parameter Settings section
Digital Input Pin	SW2	This pin is used to generate interrupts when the user button (**SW2**) is pressed.	**[General tab]** Uncheck HW connection Drive mode: Resistive Pull Up
Digital Output pin	Disconnect_LED Advertising_LED Connected_LED	These GPIOs are configured as firmware-controlled digital output pins that control LEDs.	**[General tab]** Uncheck HW connection Drive mode: Strong Drive
SysInt	SW2_Int	This Component is configured to extract interrupts from GlobalSignal.	**[Basic tab]** DeepSleepCapable = true
GSRef	GlobalSignal	This Component is used to detect if any of the interrupt enabled pins triggered an interrupt. It is a separate resource from the dedicated port interrupts, and it has the ability to wake up the chip from deep-sleep mode	**[Basic tab]** Global signal name: HWCombined Port Interrupt (AllPortInt)
SysInt	Timer_Int	This Component is configured to extract interrupts from MCVDT.	**[Basic tab]** DeepSleepCapable = true
Multi-Counter Watchdog	MCWDT	This Component is used is used for LED blinking and for counting profile specific timeouts	**[General tab]** Counter 0: Mode = Interrupt ClearOnMatch = ClearOnMatch
UART (SCB)	UART_DEBUG	This Component is used to print messages on a terminal program.	Default

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

Parameter Settings

The BLE Component is configured as the Environmental Sensing Server in the GAP Peripheral role. Also, the Battery and Device Information Services are included.

Figure 11. General Settings

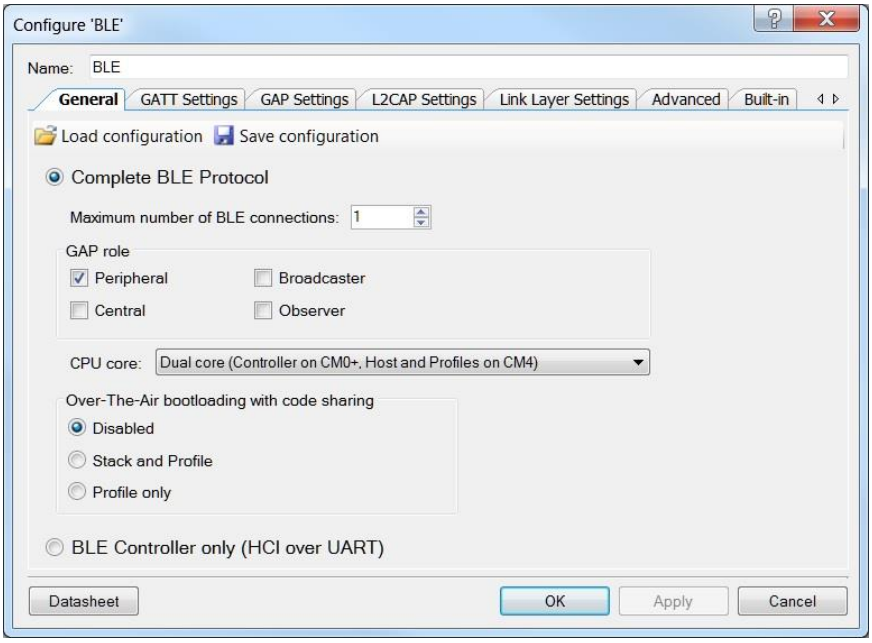


Figure 12. GATT Settings

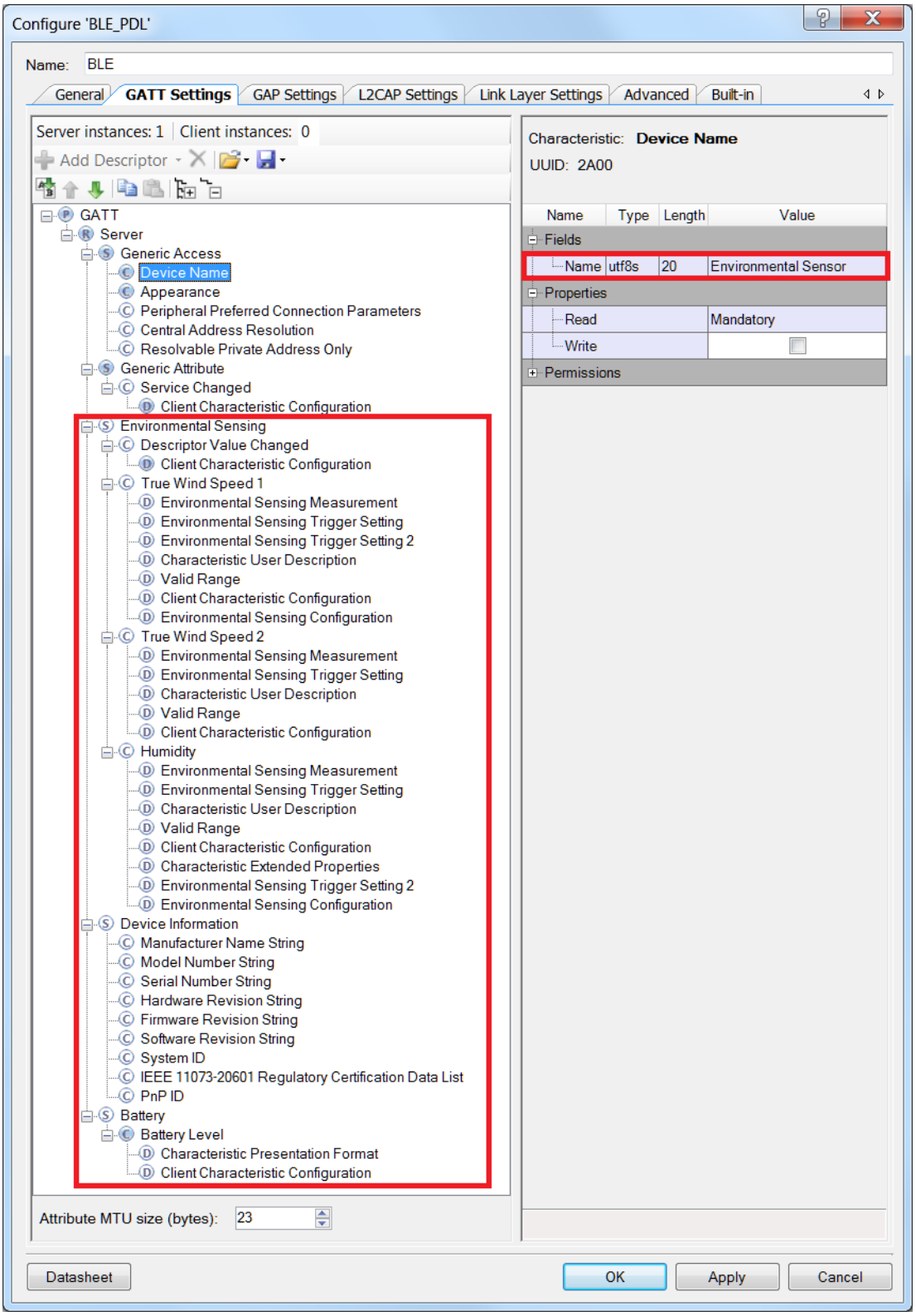


Figure 13. GAP Settings

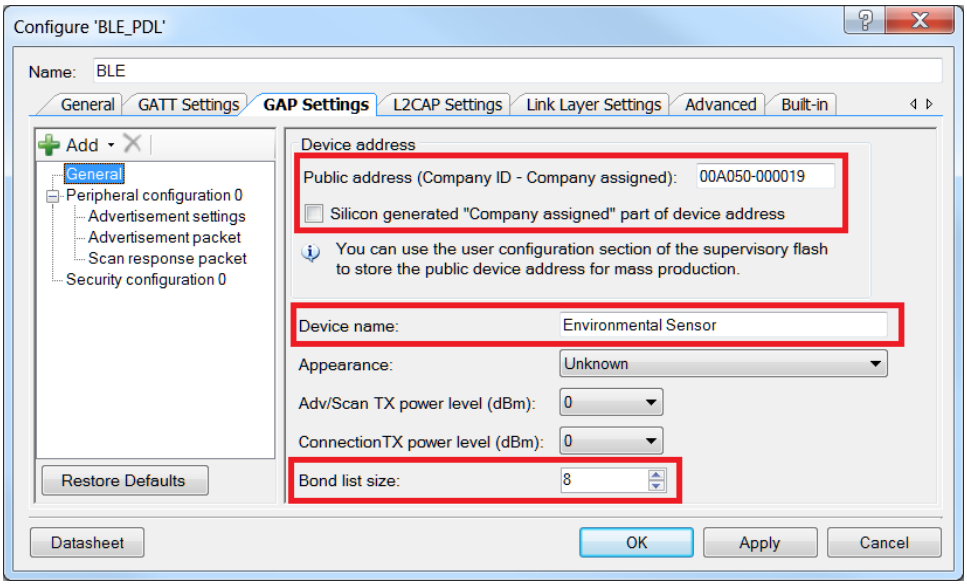


Figure 14. GAP Settings > Advertisement Setting

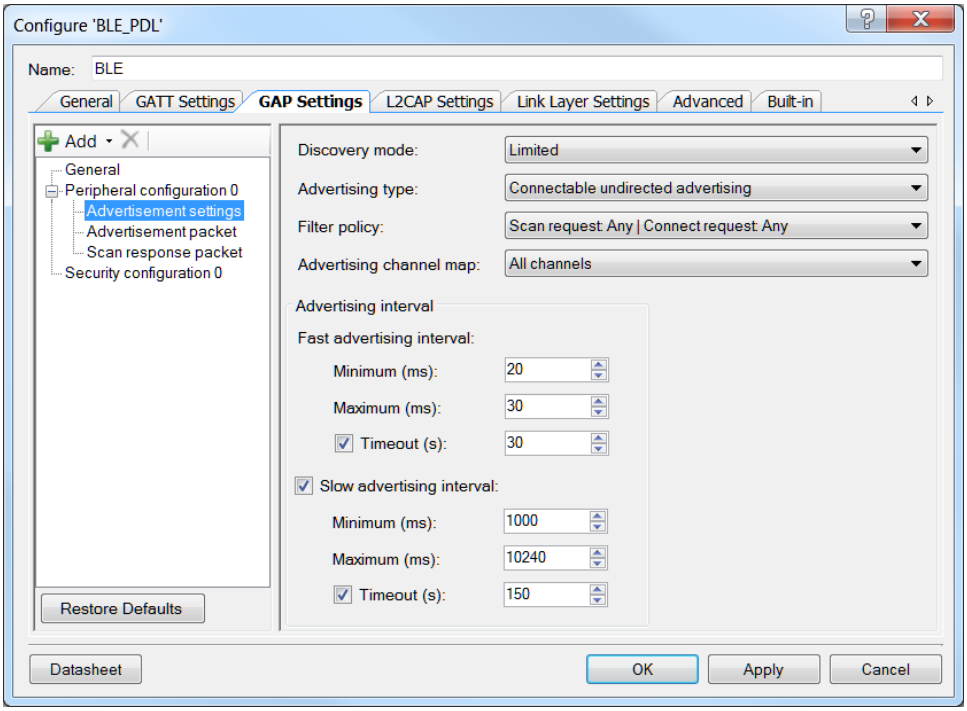
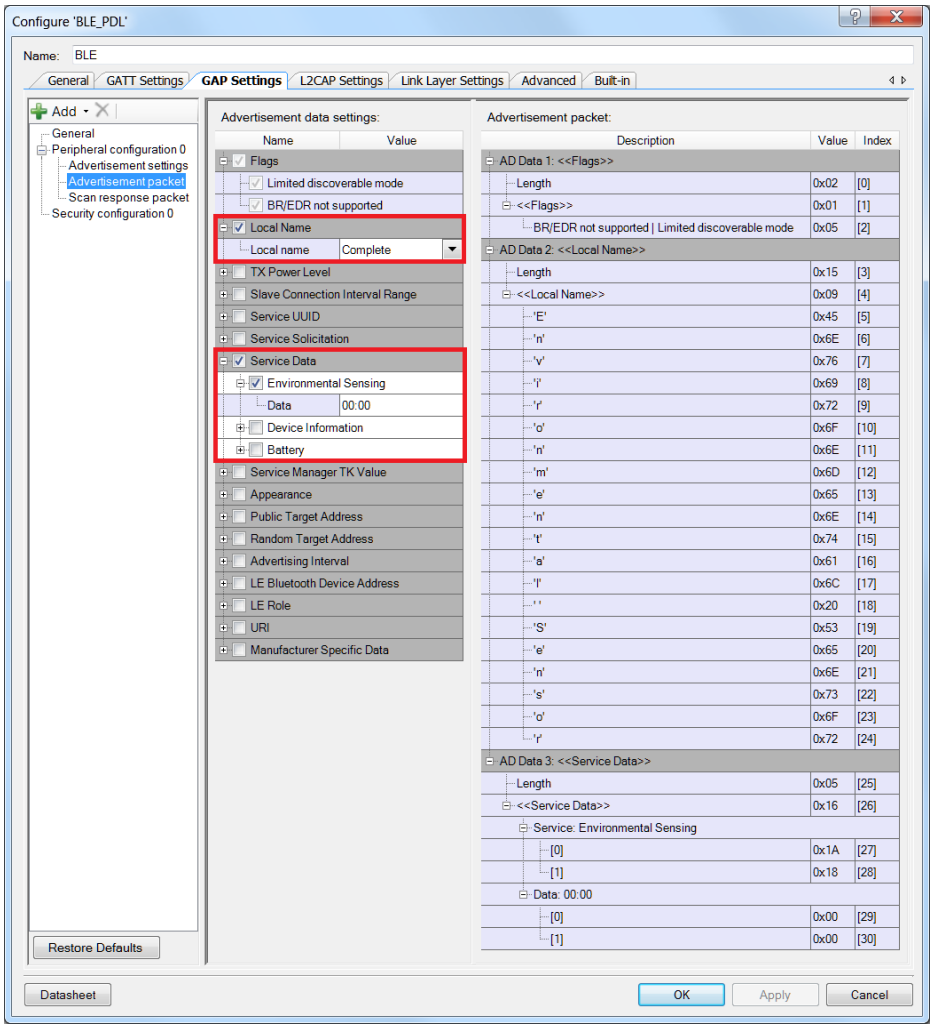
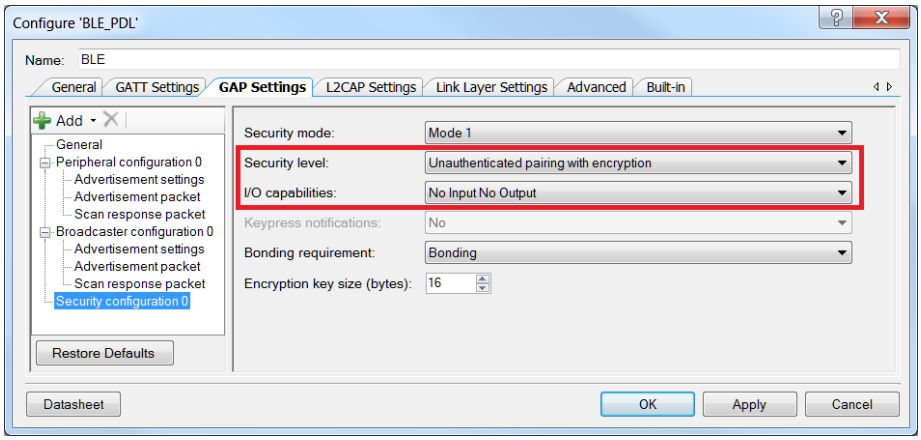


Figure 15. GAP Settings > Advertisement Packet



The Scan response packet settings are also configured to include the Local Name and all the service UUIDs into the Scan response packet.

Figure 16. GAP Settings > Security Configuration



Switching the CPU Cores Usage

This section describes how to switch between different CPU cores usage (Single core/ Dual core) in the BLE PDL examples.

The BLE component has the CPU Core parameter that defines the cores usage. It can take the following values:

- Single core (Complete Component on CM0+) – only CM0+ core will be used.
- Single core (Complete Component on CM4) – only CM4 core will be used.
- Dual core (Controller on CM0+, Host and Profiles on CM4) – both cores will be used: CM0+ for the Controller and CM4 for the Host and Profiles.

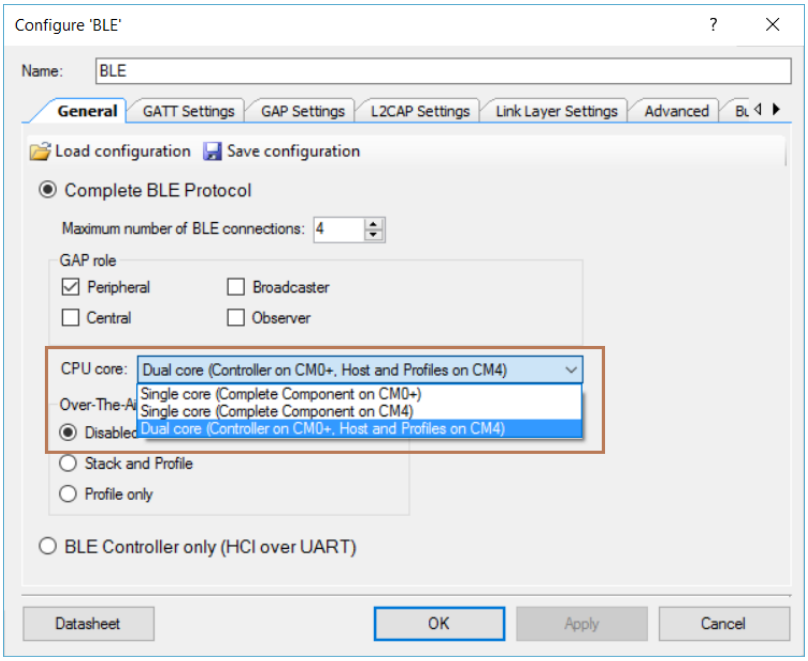
The BLE example structure allows easy switching between different CPU cores options. Important to remember:

- All application host-files must be run on the host core.
- The BLESS interrupt must be assigned to the core where the controller runs.
- All additional interrupts (SW2, MCWDT, etc.) used in the example must be assigned to the host core.

Steps for switching the CPU Cores usage:

1. In the BLE customizer **General** tab, select appropriate CPU core option.

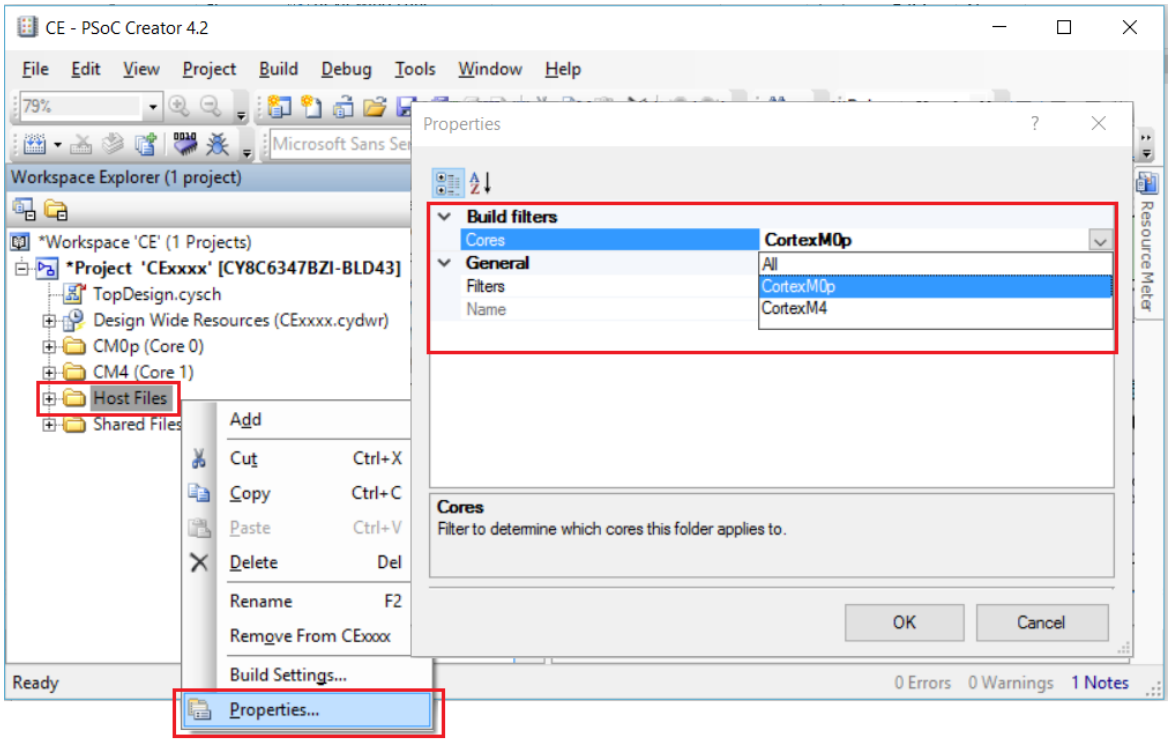
Figure 17. Select CPU Core



2. Identify the core on which host files will run. In the workspace explorer panel, right click **Host Files**, choose **Properties**. Set the **Cores** property corresponding to the CPU core chosen in step 1, as shown in Figure 18.

   - for Single core (Complete Component on CM0+) option – CM0+
   - for Single core (Complete Component on CM4) option – CM4
   - for Dual core (Controller on CM0+, Host and Profiles on CM4) option – CM4

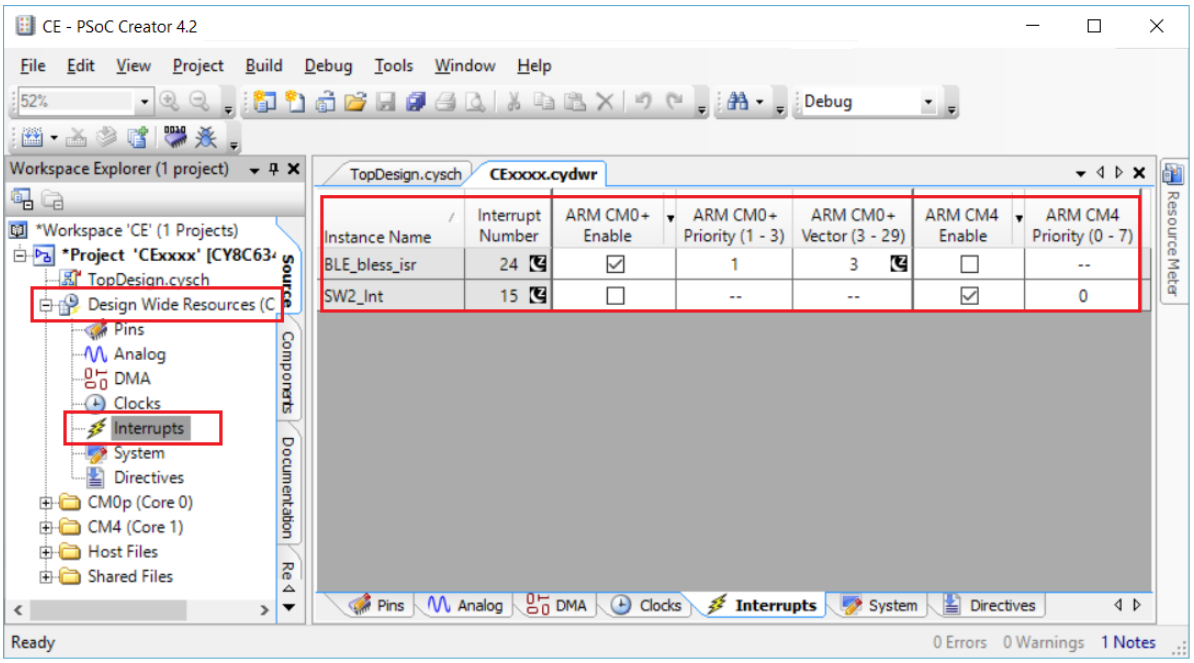
Figure 18. Change Core Properties



3. Assign the BLE_bless_isr and other peripheral (button – SW2, timer(s) etc.) interrupts to appropriate core in DWR-> interrupts tab:

   - for **Single core (Complete Component on CM0+)** option: BLE_bless_isr and peripheral interrupts on **CM0+**
   - for **Single core (Complete Component on CM4)** option: BLE_bless_isr and peripheral interrupts on **CM4**
   - for **Dual core (Controller on CM0+, Host and Profiles on CM4)** option: BLE_bless_isr interrupt on **CM0+,** other peripheral interrupts on **CM4**

Figure 19. Assign Interrupts



Reusing This Example

This example is designed for the CY8CKIT-062-BLE pioneer kit. To port the design to a different PSoC 6 MCU device and/or kit, change the target device using the Device Selector and update the pin assignments in the Design Wide Resources Pins settings as needed.

Related Documents

The following table lists all relevant application notes, code examples, knowledge base articles, device datasheets and Component datasheets.

Table 3. Related Documents

Application Notes		
AN210781	Getting Started with PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity	Describes PSoC 6 BLE, and how to build a basic code example.
AN215656	PSoC 6 MCU Dual-CPU System Design	Presents the theory and design considerations related to this code example.
Software and Drivers		
CySmart – Bluetooth® LE Test and Debug Tool		CySmart is a Bluetooth® LE host emulation tool for Windows PCs. The tool provides an easy-to-use Graphical User Interface (GUI) to enable the user to test and debug their Bluetooth LE peripheral applications.
PSoC Creator Component Datasheets		
Bluetooth Low Energy (BLE_PDL) Component		The Bluetooth Low Energy (BLE_PDL) Component provides a comprehensive GUI-based configuration window to facilitate designing applications requiring BLE connectivity.
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: CE217636 - BLE Environmental Sensing Profile with PSoC 6 MCU with BLE Connectivity

Document Number: 002-17636

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	6090347	NPAL	03/15/2018	New spec

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 Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

All other trademarks or registered trademarks referenced herein are the property of their respective owners.

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

© Cypress Semiconductor Corporation, 2018. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.