BLE Battery Level Example Project

1.0

Features

- Custom profile in Server role operation
- Battery Service usage
- Measures battery level voltage and converts it to percent
- Simulates battery charging
- DeepSleep mode demonstration
- LED status indication

General Description

This project demonstrates measurements of the battery voltage using PSoC 4 BLE/PRoC BLE's internal ADC and notifies the BLE central device of any change in the battery voltage.

Development Kit Configuration

1. Connect J2 pin P3[0] to J3 pin VREF.
2. Build the project and program the hex file into the CY8CKIT-042 BLE Pioneer Kit.

Project Configuration

The example project consists of the following components: an ADC_SAR_Seq, BLE, digital output pin, digital input pin, analog input pin. The ADC_SAR_Seq is used to measure the battery voltage. The output pins are used to reflect the line signal output on the LED. The input pin is configured to the resistive pull up mode and used to wake device from low power hibernate mode. The top design schematic is shown in **Figure 1**.

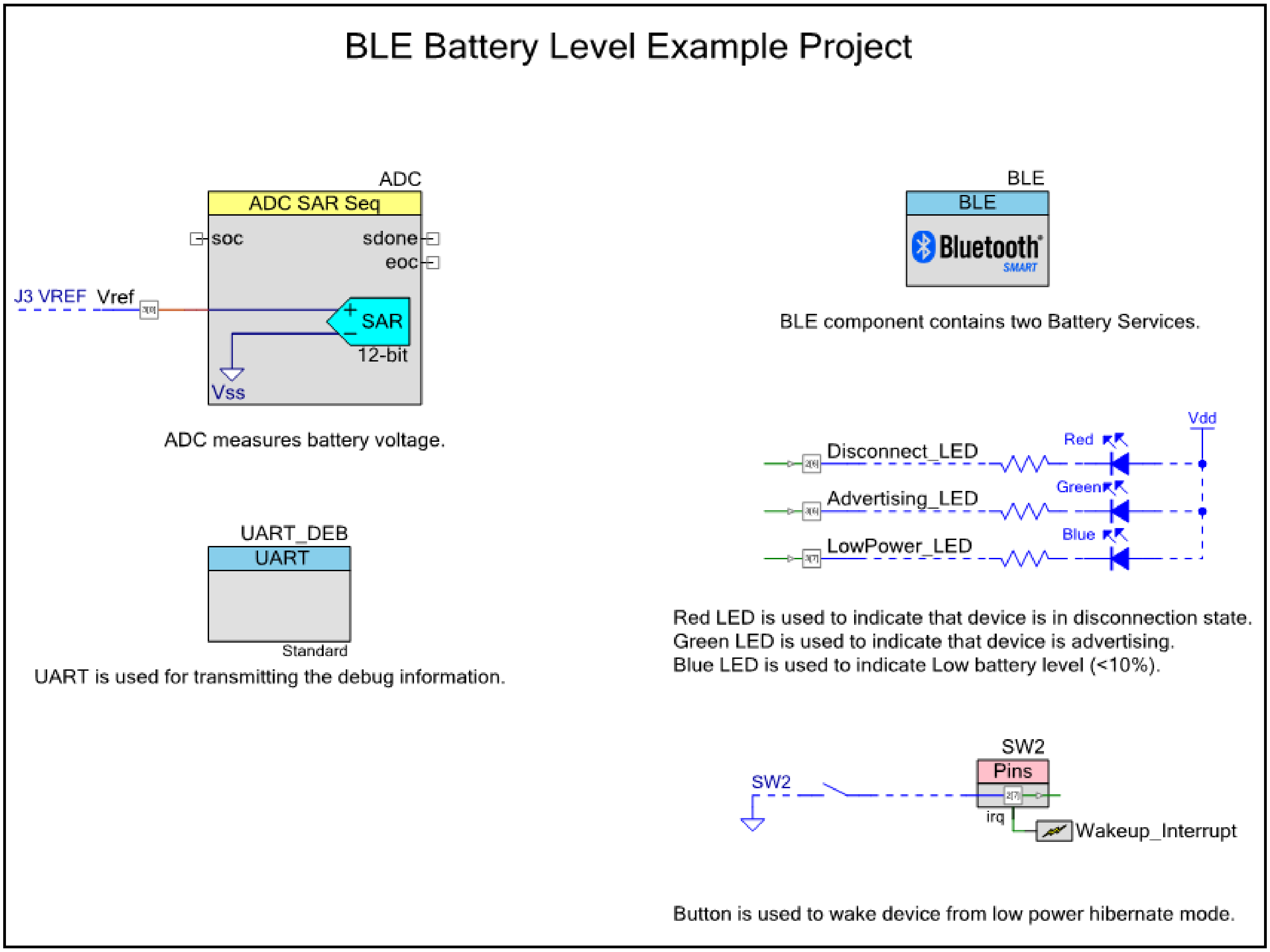
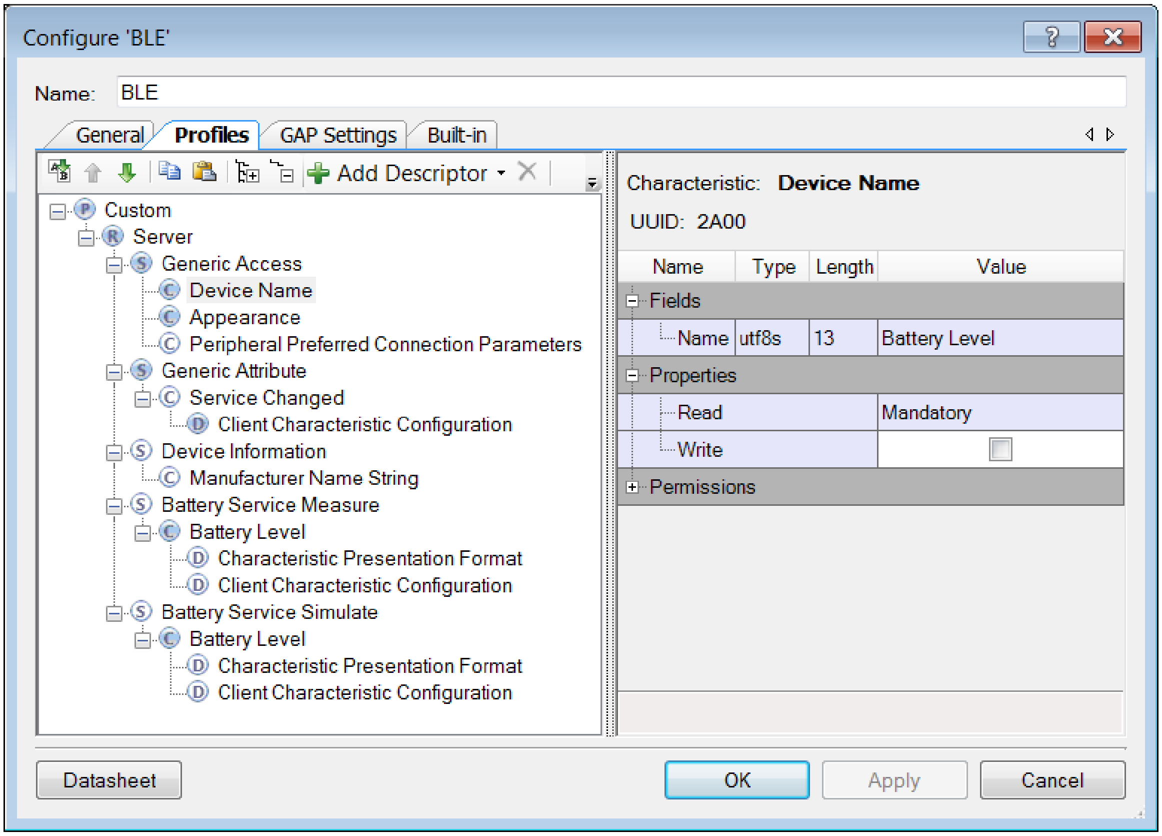

BLE Battery Level Example Project
ADC
ADC SAR Seq
soc
sdone
eoc
J3 VREF
Vref
SAR
12-bit
Vss
ADC measures battery voltage.
BLE
BLE
Bluetooth SMART
BLE component contains two Battery Services.
Vdd
Red
Green
Blue
Disconnect_LED
Advertising_LED
LowPower_LED
Red LED is used to indicate that device is in disconnection state.
Green LED is used to indicate that device is advertising.
Blue LED is used to indicate Low battery level (<10%).
UART_DEB
UART
Standard
UART is used for transmitting the debug information.
SW2
Pins
SW2
irq
Wakeup_Interrupt
Button is used to wake device from low power hibernate mode.

Figure 1. Top design schematic

The Vref analog input pin is locked to P3[0]. The ADC is configured for a single ended measurement with a sample rate of 1.60 kbps.

The BLE component is configured as Custom Profile in Peripheral Role. Device Information Service added with the Manufacture Name String characteristic. There are two Battery Services. The first one (Battery Service Measure) sends a measured battery level, the second is for software simulation of the battery level (Battery Service Simulate).

The GATT settings:



Name: BLE

- Custom
  - Server
    - Generic Access
      - Device Name
      - Appearance
      - Peripheral Preferred Connection Parameters
    - Generic Attribute
      - Service Changed
        - Client Characteristic Configuration
    - Device Information
      - Manufacturer Name String
    - Battery Service Measure
      - Battery Level
        - Characteristic Presentation Format
        - Client Characteristic Configuration
    - Battery Service Simulate
      - Battery Level
        - Characteristic Presentation Format
        - Client Characteristic Configuration

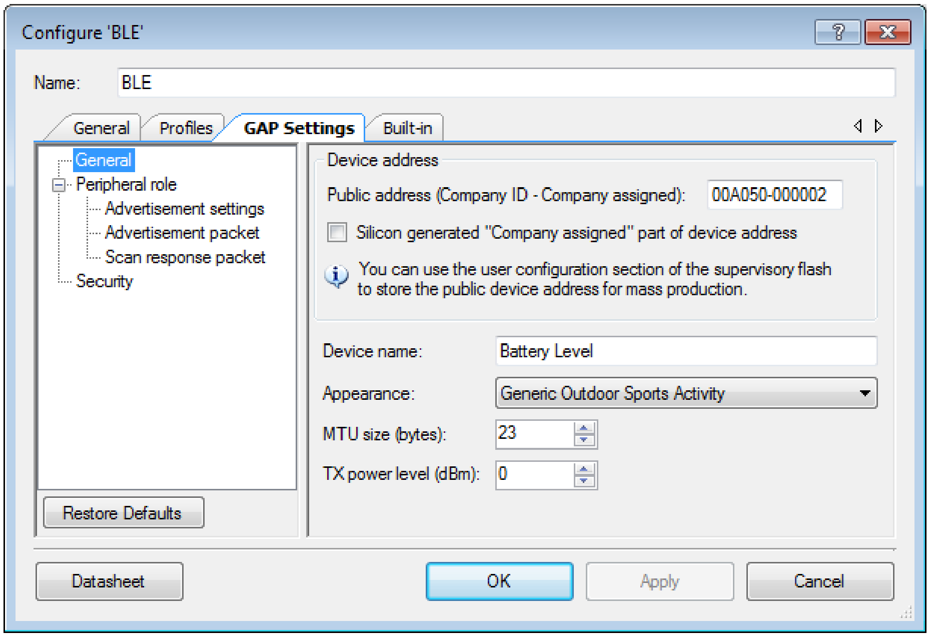
Characteristic: **Device Name**

UUID: 2A00

Name	Type	Length	Value
Fields			
Name	utf8s	13	Battery Level
Properties			
Read			Mandatory
Write			□
Permissions			

Figure 2. GATT settings

The GAP settings:



Name: BLE

Device address

Public address (Company ID - Company assigned): 00A050-000002

□ Silicon generated "Company assigned" part of device address

You can use the user configuration section of the supervisory flash to store the public device address for mass production.

Device name: Battery Level

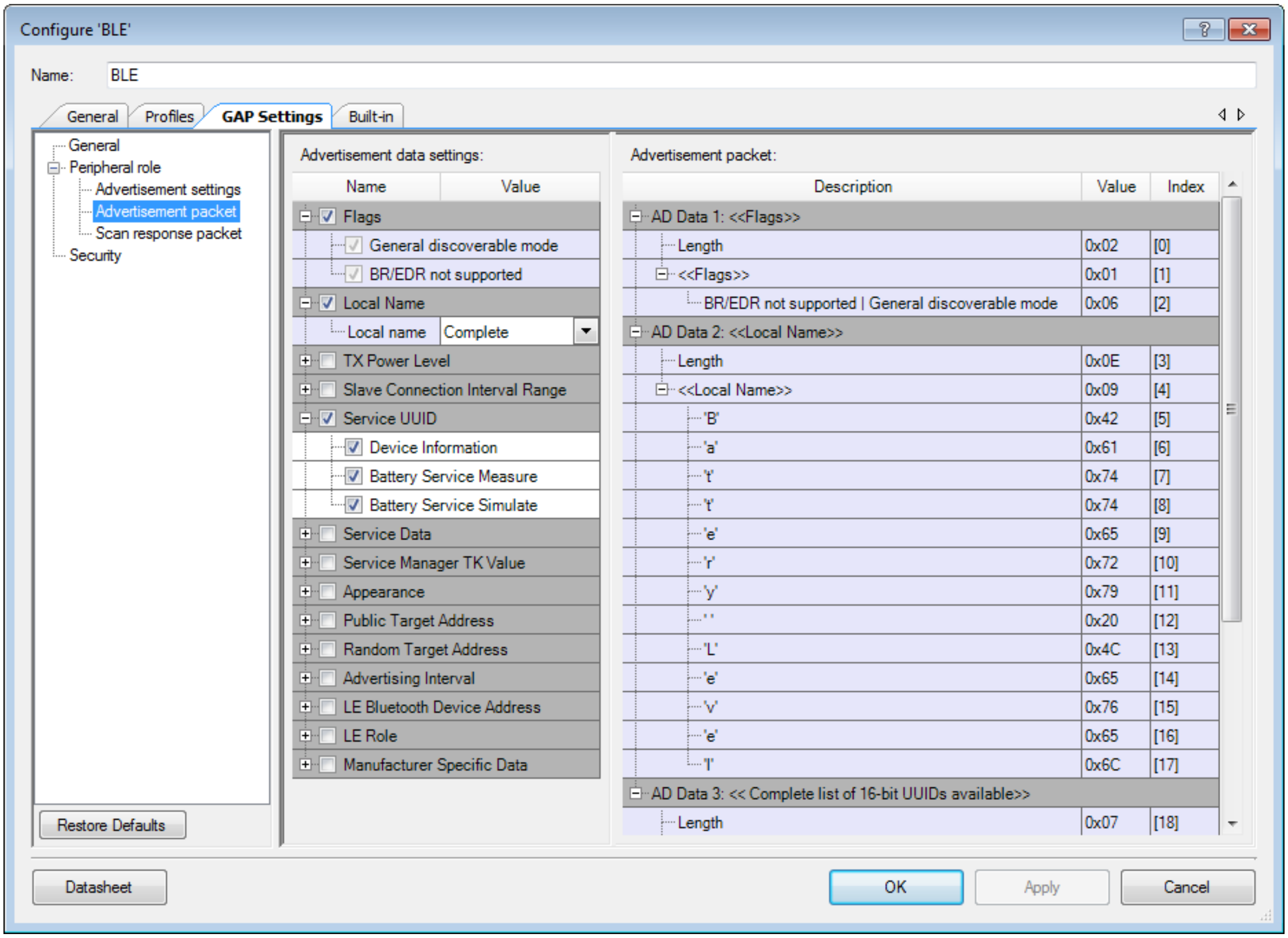
Appearance: Generic Outdoor Sports Activity

MTU size (bytes): 23

TX power level (dBm): 0

Figure 3. GAP settings

The Advertisement packet:



Name: BLE

Advertisement data settings:

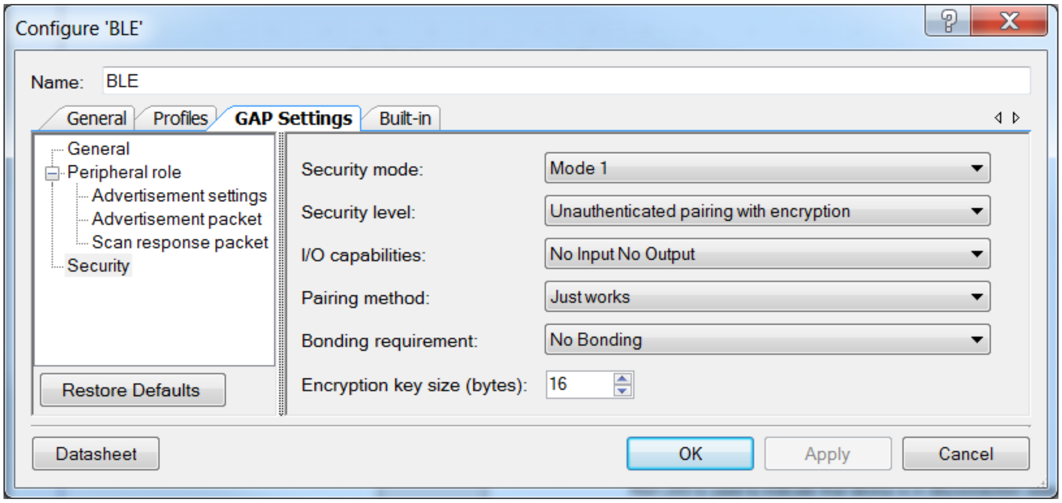
Name	Value
☑ Flags	
☑ General discoverable mode	
☑ BR/EDR not supported	
☑ Local Name	
Local name	Complete
□ TX Power Level	
□ Slave Connection Interval Range	
☑ Service UUID	
☑ Device Information	
☑ Battery Service Measure	
☑ Battery Service Simulate	
□ Service Data	
□ Service Manager TK Value	
□ Appearance	
□ Public Target Address	
□ Random Target Address	
□ Advertising Interval	
□ LE Bluetooth Device Address	
□ LE Role	
□ Manufacturer Specific Data	

Advertisement packet:

Description	Value	Index
AD Data 1: <<Flags>>		
Length	0x02	[0]
<<Flags>>	0x01	[1]
BR/EDR not supported | General discoverable mode	0x06	[2]
AD Data 2: <<Local Name>>		
Length	0x0E	[3]
<<Local Name>>	0x09	[4]
'B'	0x42	[5]
'a'	0x61	[6]
't'	0x74	[7]
't'	0x74	[8]
'e'	0x65	[9]
'r'	0x72	[10]
'y'	0x79	[11]
' '	0x20	[12]
'L'	0x4C	[13]
'e'	0x65	[14]
'v'	0x76	[15]
'e'	0x65	[16]
'l'	0x6C	[17]
AD Data 3: << Complete list of 16-bit UUIDs available>>		
Length	0x07	[18]

Figure 4. GAP settings -> Advertisement packet

The security settings:



Name: BLE

Security mode: Mode 1

Security level: Unauthenticated pairing with encryption

I/O capabilities: No Input No Output

Pairing method: Just works

Bonding requirement: No Bonding

Encryption key size (bytes): 16

Figure 5. Security settings

Project Description

This project demonstrates functionality of the Battery Service of the BLE component.

The reference to the ADC in PSoC 4 can be either internal 1.024V or VDD. A simple method to measure VDD in PSoC4 would be to use a resistive divider and scale the VDD to 1.024V and set the ADC reference to internal 1.024V. This project measures the battery voltage which is equal to the VDD voltage without an external resistive divider.

When VREF is selected as reference to the ADC, and if the reference bypass is enabled, this will bring out VREF (through an internal series resistance).

Following is the procedure to measure VDD that exploits this feature.

- Make sure that J2 pin P3[0] is connected to J3 pin VREF.
- Set the reference of ADC to VREF and enable a bypass. Keep the bypass enabled for a short time (25 ms in this test) and the external capacitor is charged to VREF.
- Change the reference of the ADC to VDD and measure the voltage on P3[0].
- Calculate the VDD voltage using the formula:

  VDD = (1.024 * Full Scale Counts) / ADC Counts P3[0]

  where Full Scale Counts = 2047. With the ADC configured for a single ended measurement, the effective resolution is 11 bits and the full scale count is 2047.

Right after the device is started it performs the BLE component initialization. Two callback functions are required in this project for BLE operation. One callback function (AppCallBack()) is required to receive generic events from BLE Stack. The other callback function (BasCallBack()) is required for receiving events from the BAS services. CyBle_GappStartAdvertisement() API is called after CYBLE_EVT_STACK_ON event to start advertising with the packet shown in **Figure** 4.

To indicate that the device is advertising, the green LED is blinking. The red LED will be lighted on after disconnection to indicate that no Client is connected to the device. When a Client is connected successfully, both red and green LEDs will turn off. When the measured battery voltage drops below 10% limit, the blue LED will be on.

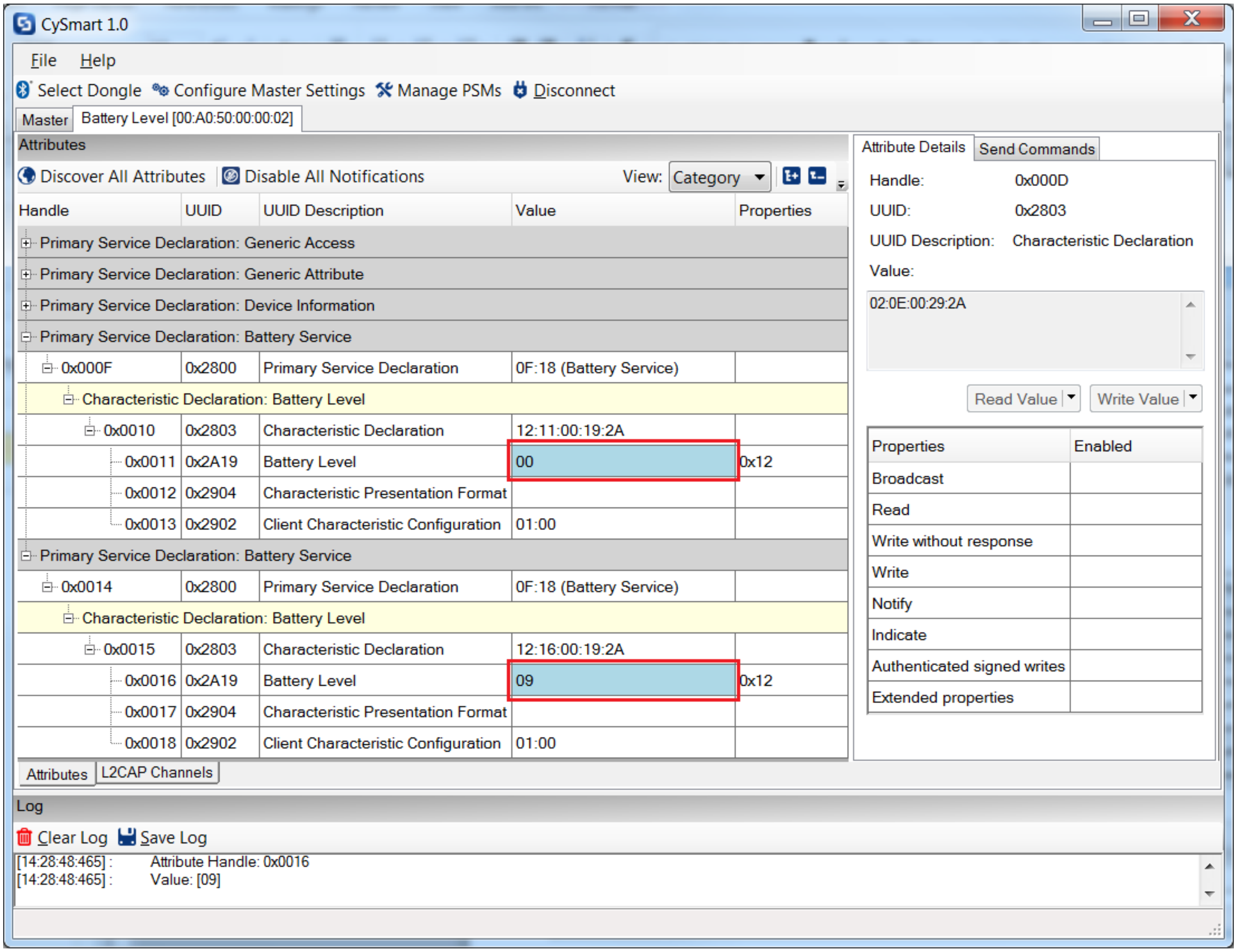
Expected Results

You can use CySmart app on a Windows PC, Android or iOS BLE-compatible device as client for connection to Battery Service.

To use CySmart Windows application as Battery Service client:

- Connect the CySmart BLE dongle to a USB port on the PC.
- Launch CySmart app and select connected dongle in dialog window.
- Reset development kit to start advertising by pressing SW1 button.

- Click **Start Scan** button to discover available devices.
- Select **Battery Level** in the list of available devices and connect to it.
- Click **Pair**, then **Discover All Attributes**, then **Read All Characteristics**, and finally **Enable All Notifications** in CySmart app.
- Notice values of Battery Level descriptors:



Handle	UUID	UUID Description	Value	Properties
Primary Service Declaration: Generic Access				
Primary Service Declaration: Generic Attribute				
Primary Service Declaration: Device Information				
Primary Service Declaration: Battery Service				
0x000F	0x2800	Primary Service Declaration	0F:18 (Battery Service)	
Characteristic Declaration: Battery Level				
0x0010	0x2803	Characteristic Declaration	12:11:00:19:2A	
0x0011	0x2A19	Battery Level	00	0x12
0x0012	0x2904	Characteristic Presentation Format		
0x0013	0x2902	Client Characteristic Configuration	01:00	
Primary Service Declaration: Battery Service				
0x0014	0x2800	Primary Service Declaration	0F:18 (Battery Service)	
Characteristic Declaration: Battery Level				
0x0015	0x2803	Characteristic Declaration	12:16:00:19:2A	
0x0016	0x2A19	Battery Level	09	0x12
0x0017	0x2904	Characteristic Presentation Format		
0x0018	0x2902	Client Characteristic Configuration	01:00	

Handle: 0x000D
UUID: 0x2803
UUID Description: Characteristic Declaration
Value:
02:0E:00:29:2A

Properties	Enabled
Broadcast	
Read	
Write without response	
Write	
Notify	
Indicate	
Authenticated signed writes	
Extended properties	

```
[14:28:48:465] :     Attribute Handle: 0x0016
[14:28:48:465] :     Value: [09]
```

Figure 6. CySmart Windows app

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

To use CySmart mobile app as Battery Service client:

- Launch CySmart mobile app on the Android or iOS BLE-compatible device.

- Swipe down to refresh the list of found BLE devices.
- Connect to “Battery Level” device and select “Battery Service” in service list.
- Notice the value of Battery Level.

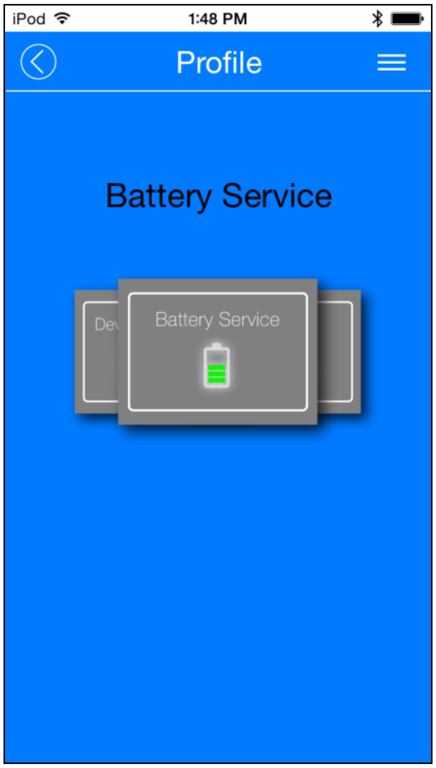


Figure 7. CySmart iOS app recognized Battery Service

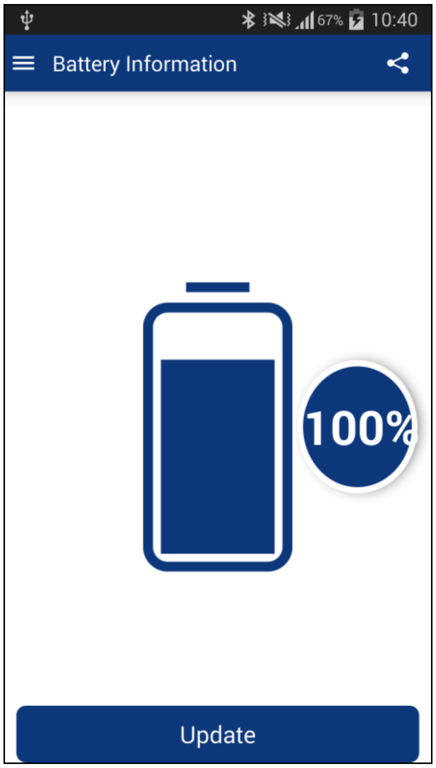


Figure 8. CySmart Android app shows battery level

Also, you can notice that the blue color LED will be on when a measured battery level is less than 10%.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.