CYBLE-022001-00 : I2C-BLE Peripheral Role

Objective

This example demonstrates how to use CYBLE-02001-00 device as I2C-BLE Peripheral device.

Overview:

This code example uses a custom BLE profile with two custom service to demonstrate the I2C-BLE peripheral functionality. One custom service is used to notify the data written by the I2C master to the client and the second one is used to update the I2C read registers when the client writes to the GATT DB of the peripheral device

Requirements:

Programming Language	**:** C (GCC 4.8.4)
Associated Parts	**:** CYBLE-022001-00
Required software	**:** PSoC Creator 3.1 SP2, Bridge Control Panel 1.12.0.2043 , PSoC Programmer 3.22.3 CySmart PC application
Required hardware	**:** CY8CKIT-042-BLE Bluetooth® Low Energy (BLE) Pioneer Kit , CYBLE-022001-EVAL, *2* wires with male header on one side and female header on other side
Optional hardware	**:** Bluetooth sniffer

Project Description:

The project source code is maintained in the GitHub location: https://github.com/cypresssemiconductorco/EZ-BLE_PRoC_Module/tree/master/Example_projects

This project demonstrates the following.

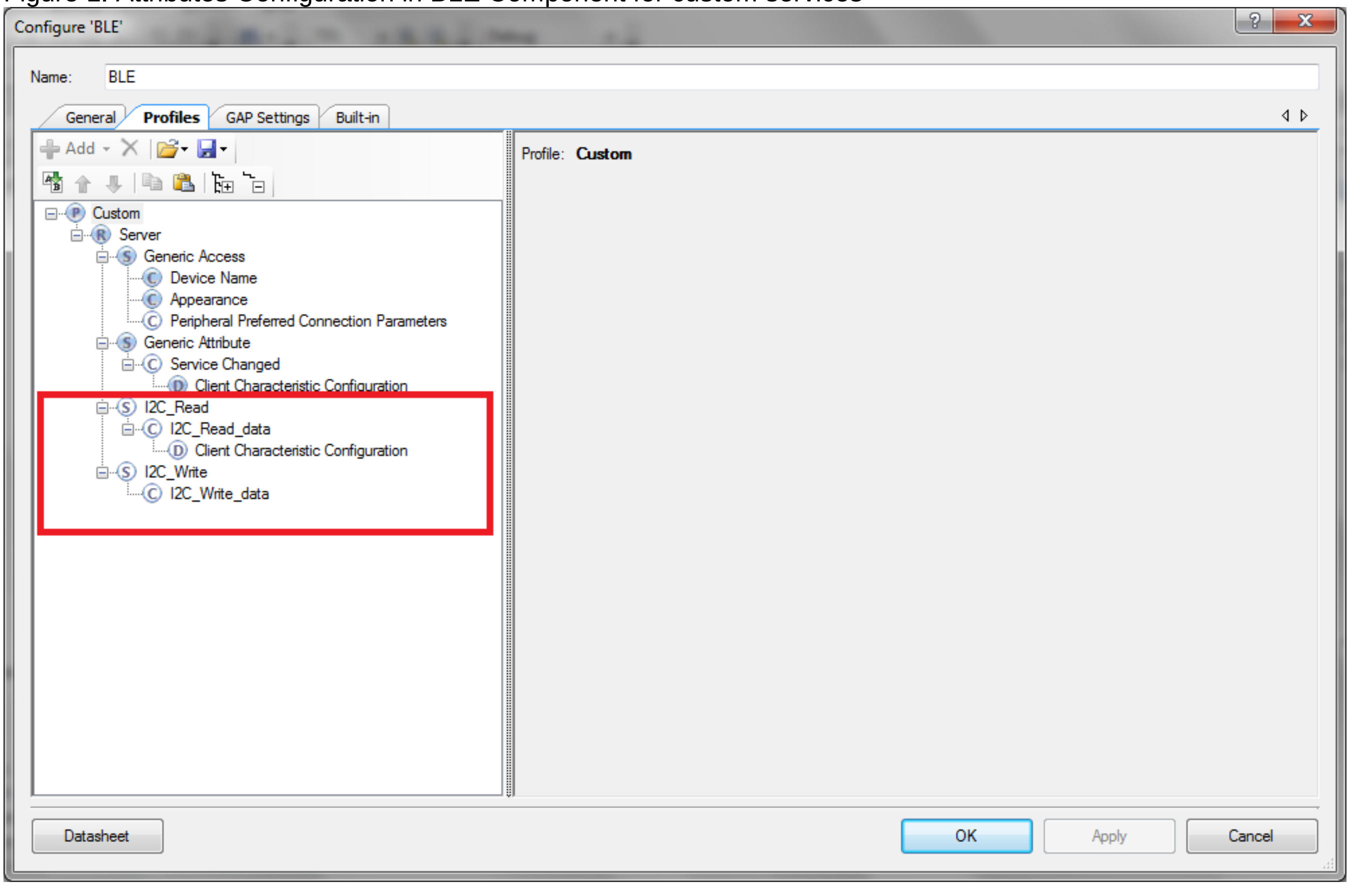
- I2C-BLE bridge implementation – Peripheral role
- Connection with any central device
- Two custom services in a single profile
- Low power implementation for coin-cell operation

The BLE profile in this project consists of two custom services: I2C_Read and I2C_Write. The I2C_Read service consists of one custom characteristics which is used to send the I2C data written by the I2C master to the client as notifications. This simulates the I2C write in case of wired I2C.

The second custom service I2C_Write consists of one custom characteristics through which the GATT client can write data to the peripheral device. The peripheral device in turn update the data written by the GATT client to the I2C read registers for the I2C master to read. This simulates the I2C slave updating its read registers for the master to read.

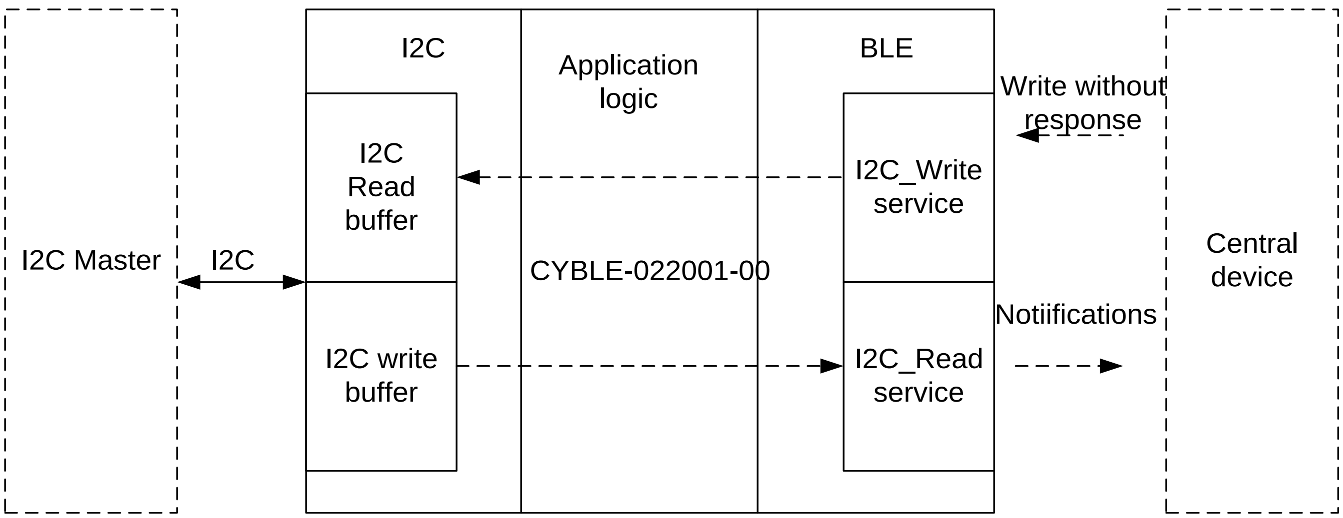
These properties for the custom service/characteristics are configured in the BLE component under the Profiles tab, as shown in Figure 1 below

Figure 1. Attributes Configuration in BLE Component for custom services



A simple block diagram of the implementation is shown in the figure 2.

Figure 2. Block diagram of the implementation



The project consist of the following files:

Main.c/.h

These files contain the main function, which is the entry point and execution of the firmware application. It also contains function definition for initialization of the system.

App_BLE.c/.h

These files contain all the macros and function definitions related to BLE communication and operation. It contains the event callback function definition that is registered with the BLE component startup and used by the component to send BLE-related events from the BLE stack to the application layer for processing. It contains a method to send notifications to the GATT client device and process the Write commands on the I2C_Read characteristic by the GATT client device.

Low_power.c/.h

These files contain the function to handle low-power mode. This function is continuously called in the main loop and is responsible for pushing the BLE hardware block (BLESS) as well as the CPU to Deep Sleep mode as much as possible. The wakeup source is either the BLE hardware block Link Layer internal timer or the interrupt from the I2C address match. This allows for very low power mode implementation and operation using a coin cell.

App_I2C.c/.h

These files contain the function to handle the I2C read and write activity.

Config.h

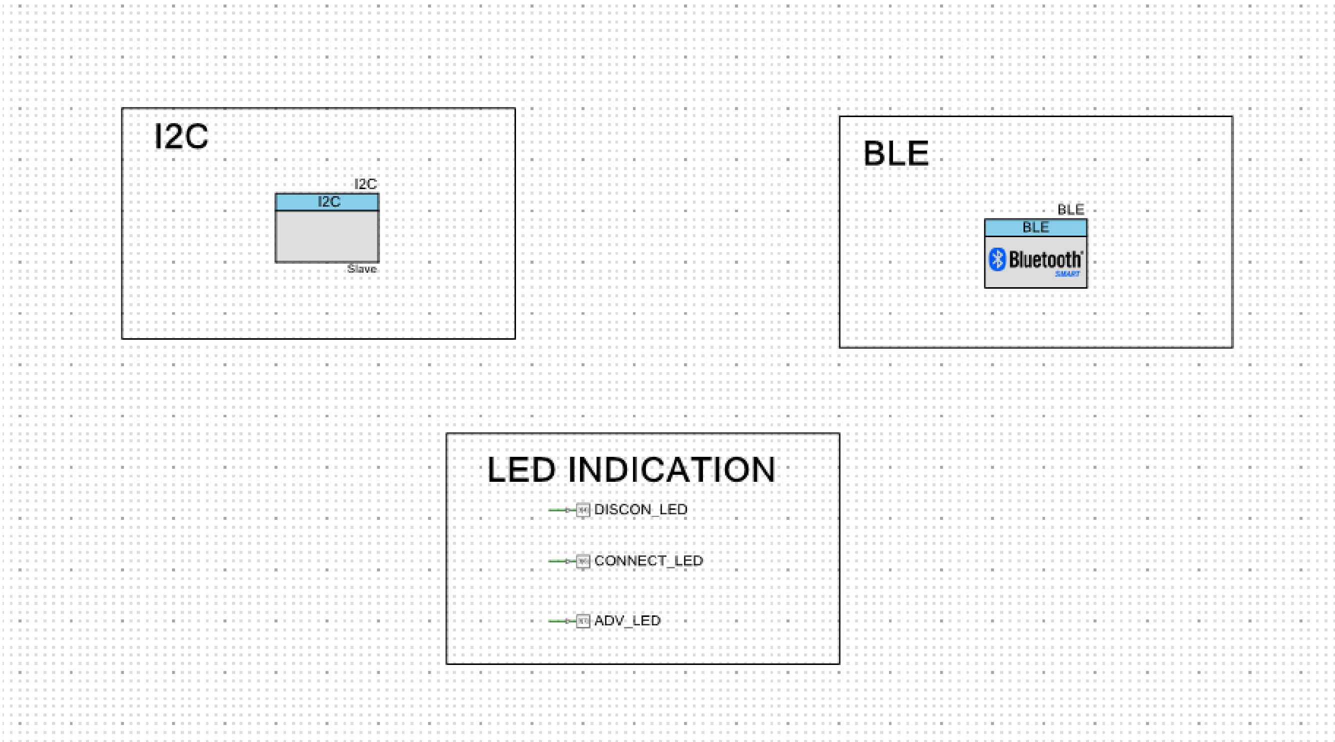
This file has macros to enable or disable – Low power mode implementation and LED indication

Additionally there are LED indications to show the state of the device.
BLUE LED – Device is in advertisement state
RED LED – Device is in disconnected state
GREEN LED – Device is in connected state

To measure the best power consumption number, disable the LED indication.

The SWD pin are configured as GPIO to get the lowest possible current consumption number.

The top design of the project is shown in the figure 3 below.
Figure 3 Top Design for I2C_BLE Bridge Project



Hardware Connections

No specific hardware connections are required for this project because all connections are hardwired on the BLE Pioneer Baseboard. Ensure that the module is placed on the baseboard correctly.
The pin assignment for this project is in **I2C_BLE_Bridge.cydwr** in the Workspace Explorer, as shown in Figure 4

Figure 4 Pin Selection for I2C_BLE Project

Alias	Name	Port	Pin	Lock
	\I2C:scl\	P5[1] TCPWM3:line_out_compl, SCB1:uart_tx, SRSS:ext_clk, SCB1:i2c_scl, SCB1:spi_clk	39	☑
	\I2C:sda\	P5[0] TCPWM3:line_out, SCB1:uart_rx, BLESS:rfctrl_extpa_en, SCB1:i2c_sda, SCB1:spi_select[0]	32	☑
	ADV_LED	P3[7] SARMUX:pads[7], TCPWM3:line_out_compl, SCB1:uart_cts, SRSS:ext_clk_lf	4	☑
	CONNECT_LED	P3[6] SARMUX:pads[6], TCPWM3:line_out, SCB1:uart_rts	14	☑
	DISCON_LED	P3[4] SARMUX:pads[4], TCPWM2:line_out, SCB1:uart_rx, SCB1:i2c_sda	12	☑

Verify Output

The project can be verified by using the CySmart Central Emulation Tool and BLE Dongle.
You can either make use of the KitProg (PSoC-5LP) on the BLE pioneer kit base board or connect an external I2C master to verify the project.
The KitProg acts as a USB-I2C bridge. It gets data from the Bridge control panel tool over USB and sends it as I2C transaction to CYBLE-022001-EVAL.

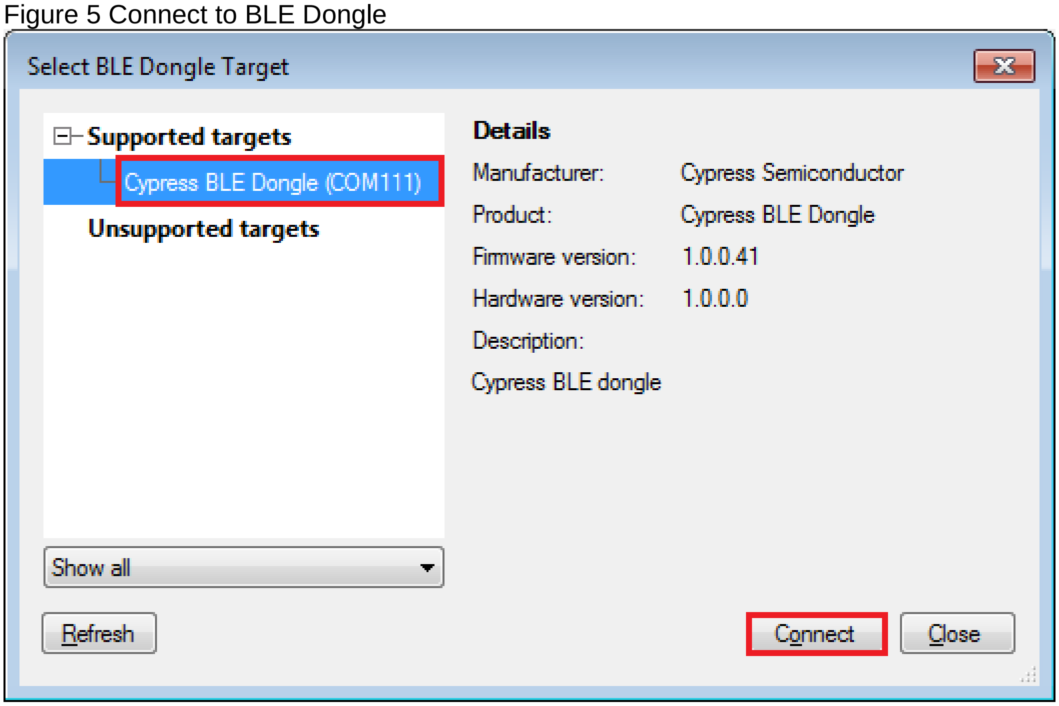
CySmart Central Emulation Tool

To verify the I2C_BLE_Bridge project using the CySmart Central Emulation Tool, follow these steps:
Note: Refer CySmart BLE Host Emulation tool to learn how to use the tool.

1. Connect the BLE Dongle to one of the USB ports on the PC and wait for the dongle to enumerate.
2. Start the CySmart Central Emulation Tool on the PC by going to **Start > All Programs > Cypress> CySmart <version> > CySmart <version>**. You will see a list of BLE Dongles connected to it.

If no BLE Dongle is found, click **Refresh**. Select the BLE Dongle and click **Connect**.

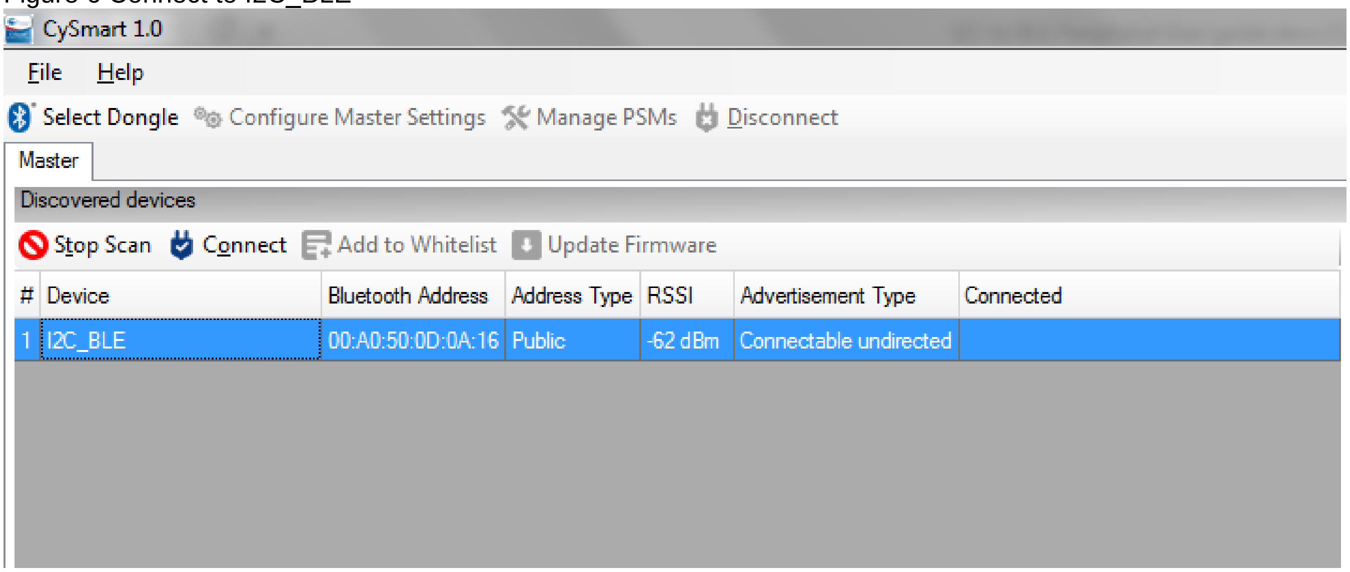
Figure 5 Connect to BLE Dongle



3. Connect the CYBLE-022001-EVAL board on the J10 and J11 headers on the BLE Pioneer Kit.
4. Connect P5.0 and P5.1 to P12.1 and P12.0 of PSoC-5LP respectively. (If you are using a different I2C Master Connect VDD, GND, P5.0 and P5.1 of CYBLE-022001-00 to VDD, GND, SDA and SCL line of the I2C master respectively)
5. Power the BLE Pioneer Kit through the USB connector **J13**.
6. Open the project in creator and build it. Program the BLE Pioneer Kit with the I2C_BLE_Bridge project.
7. After successful programming the BLUE LED will glow indicating that the device is in advertisement mode.
8. On the CySmart Central Emulation Tool, click **Start Scan** to see the list of available BLE peripheral devices.

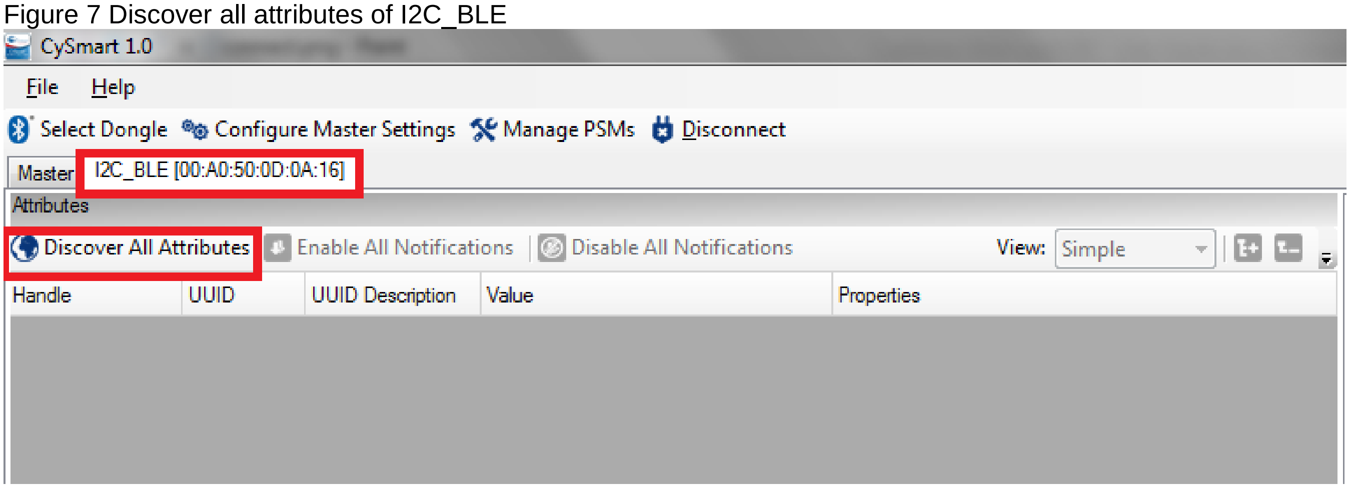
9. Double-click the **I2C_BLE** device to connect, or click **I2C_BLE** and then Click **Connect**. The GREEN LED on the device will start glowing indicating it is connected to the Client.

Figure 6 Connect to I2C_BLE



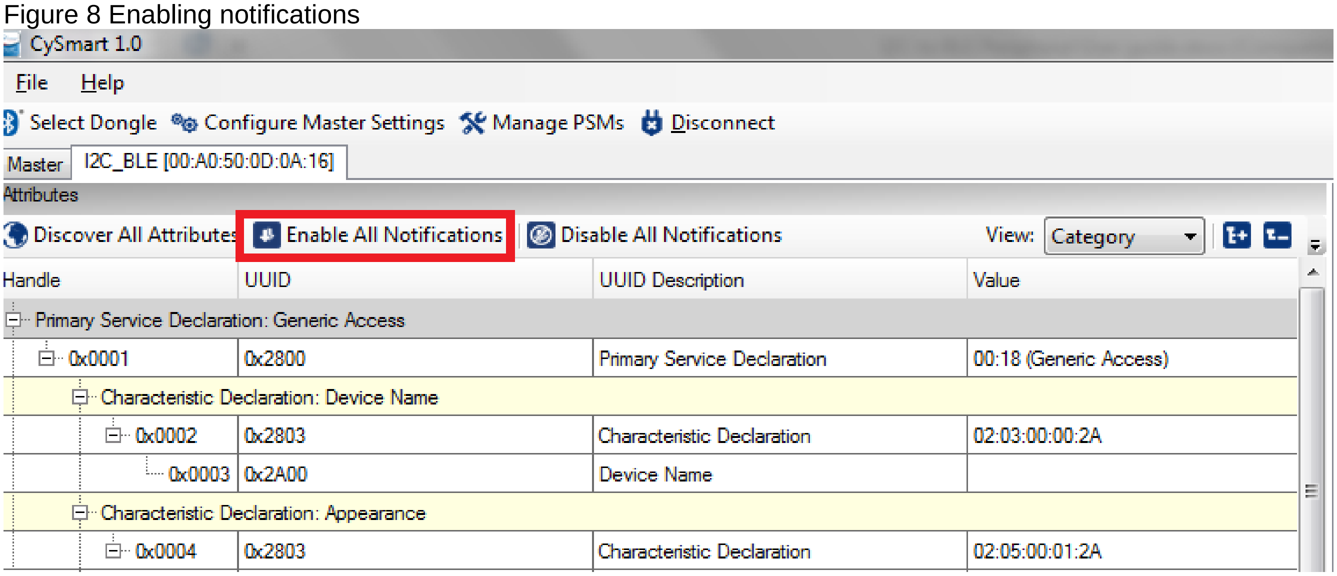
10. Click **Discover All Attributes** to find all attributes supported.

Figure 7 Discover all attributes of I2C_BLE



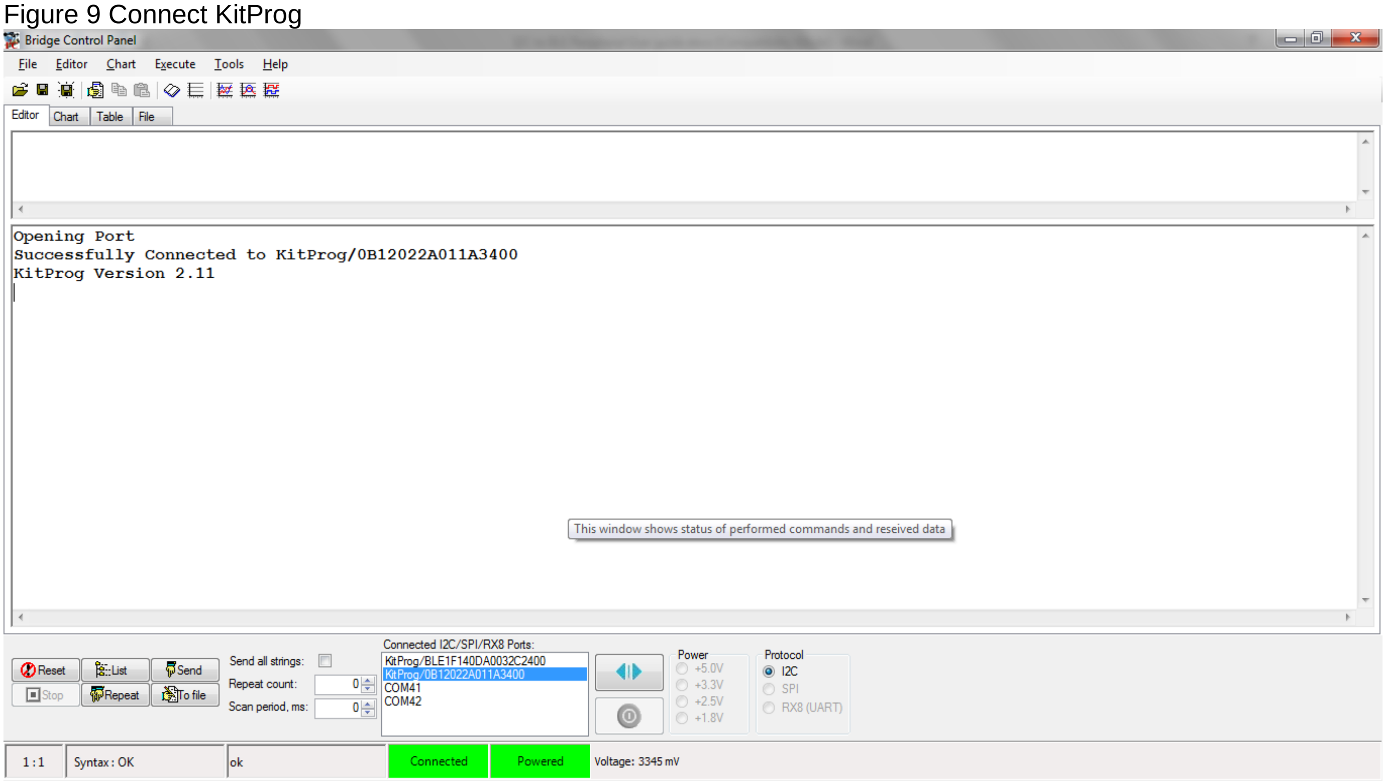
11. Click on **Enable All Notifications** to enable all notifications.

Figure 8 Enabling notifications



11. Open Bridge control panel and connect Kitprog. Make sure the KitProg is not connected anywhere else. You might have to update Kitprog firmware using PSoC programmer to the latest version to connect to Bridge control panel.

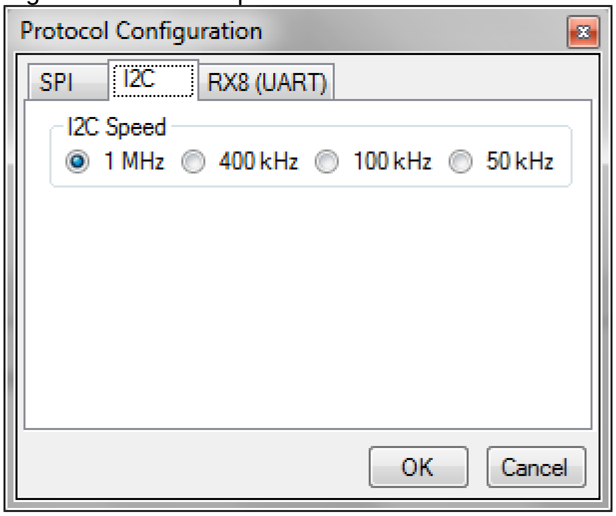
Figure 9 Connect KitProg



12. Click on Tools->Protocol configurations or press F7.

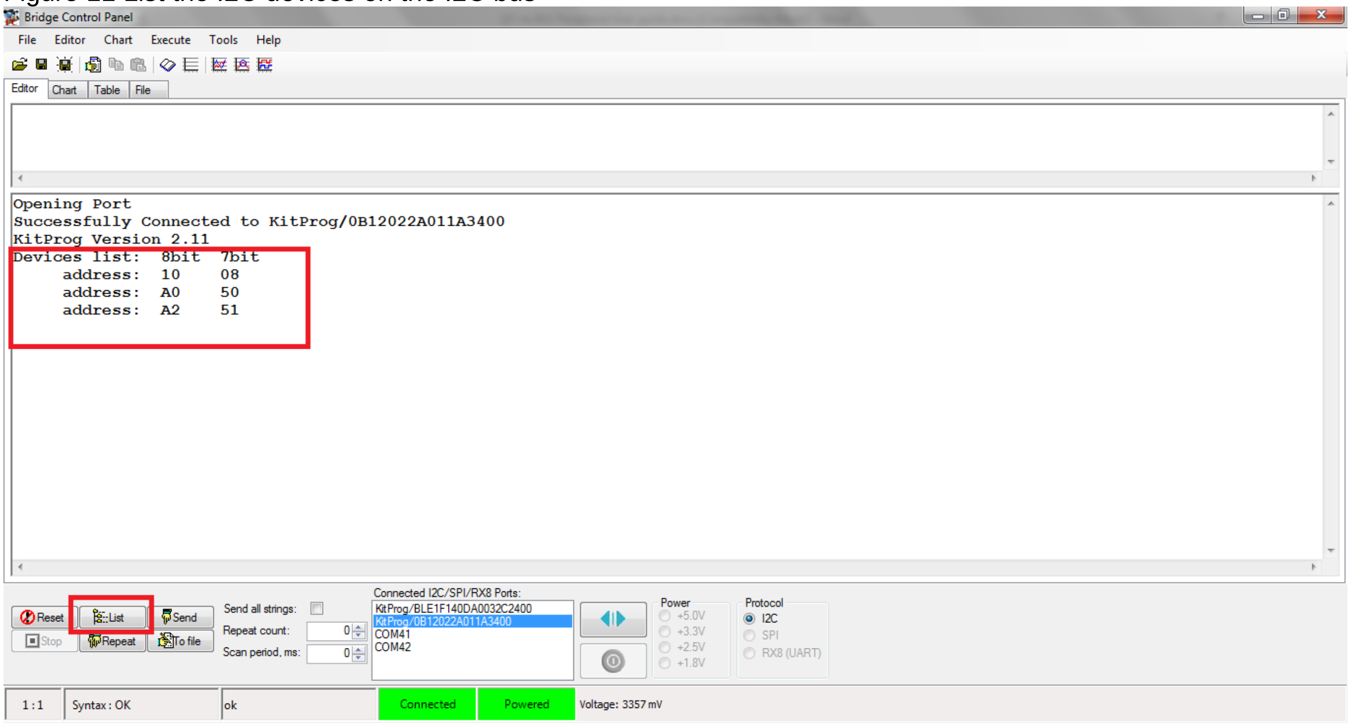
13. Set the I2C speed to 1 MHz

Figure 10 Set I2C Speed



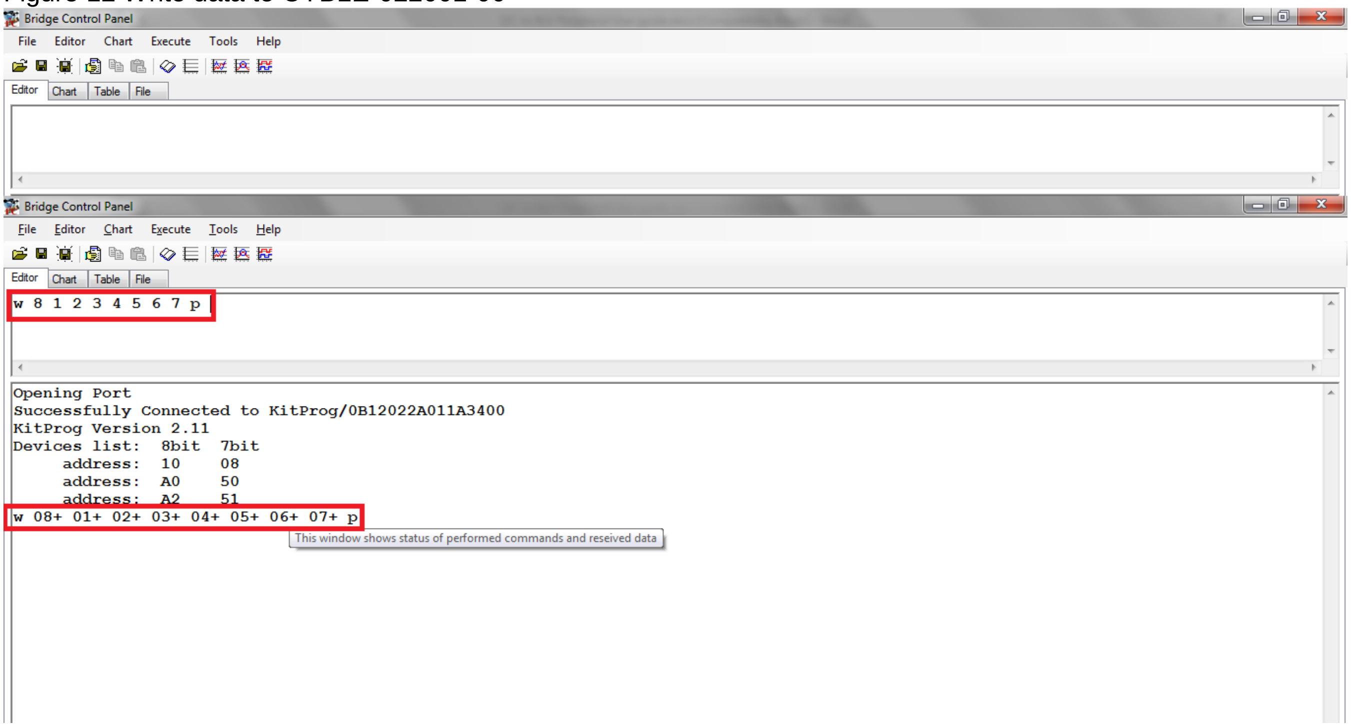
14.) Click on list and check if the slave address (0x08) of CYBLE-022001-00 is present.
Note: The FRAM chip on the pioneer kit base board also acts as I2C slave (slave address 50 and 51) and hence you will see three entries.
If you have connected the wrong KitProg you won't see the correct slave device, in that case try connecting the other KitProg to find the correct one.

Figure 11 List the I2C devices on the I2C bus



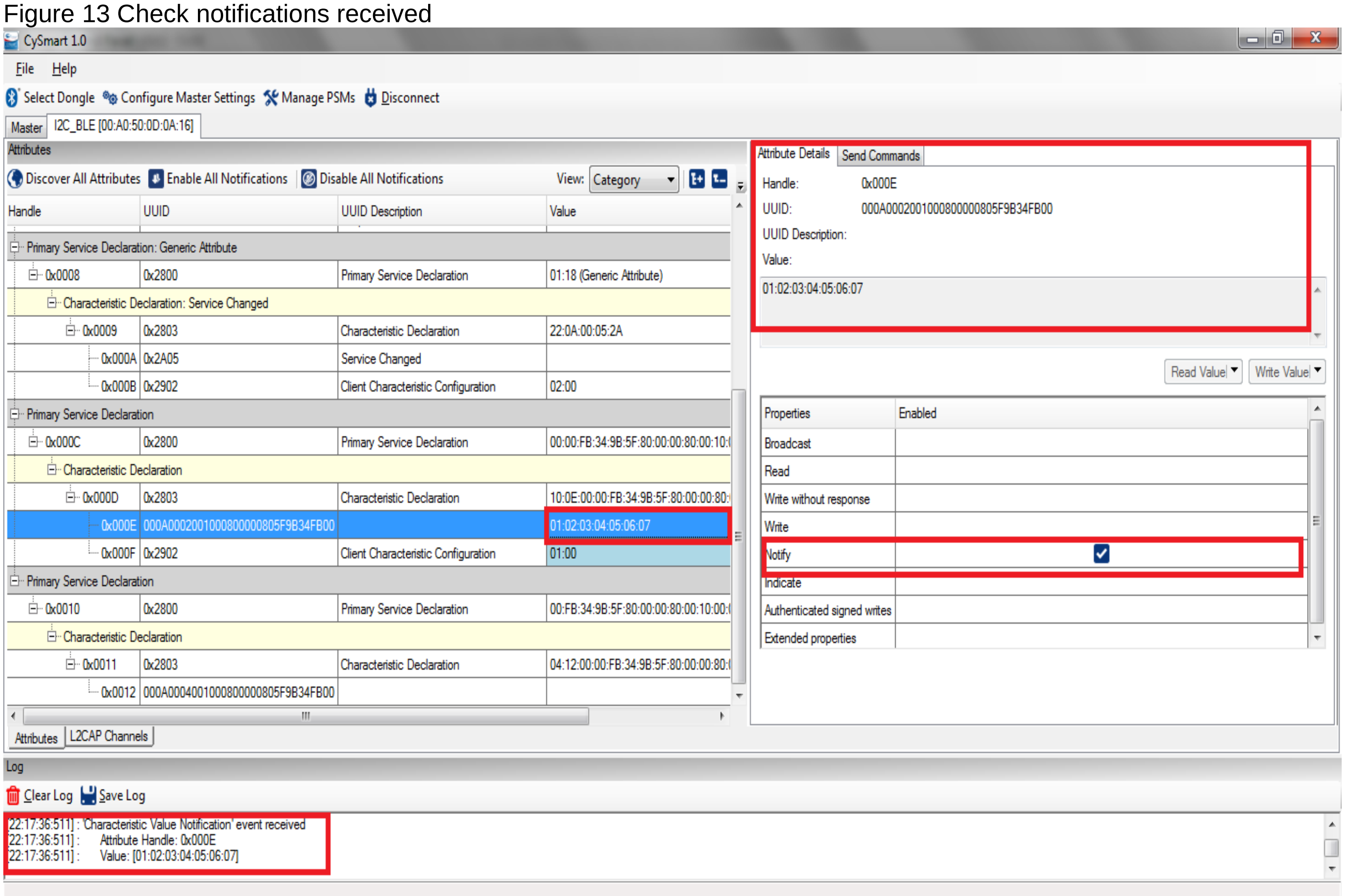
15. Write data to CYBLE-022001-00 by entering the command w 8 d1 d2 d3….dn p. w stands for write, 8 is the slave address of CYBLE-022001-00, d1 d2…dn are the data to be written to the slave. n has to be less than 61 bytes.

Figure 12 Write data to CYBLE-022001-00



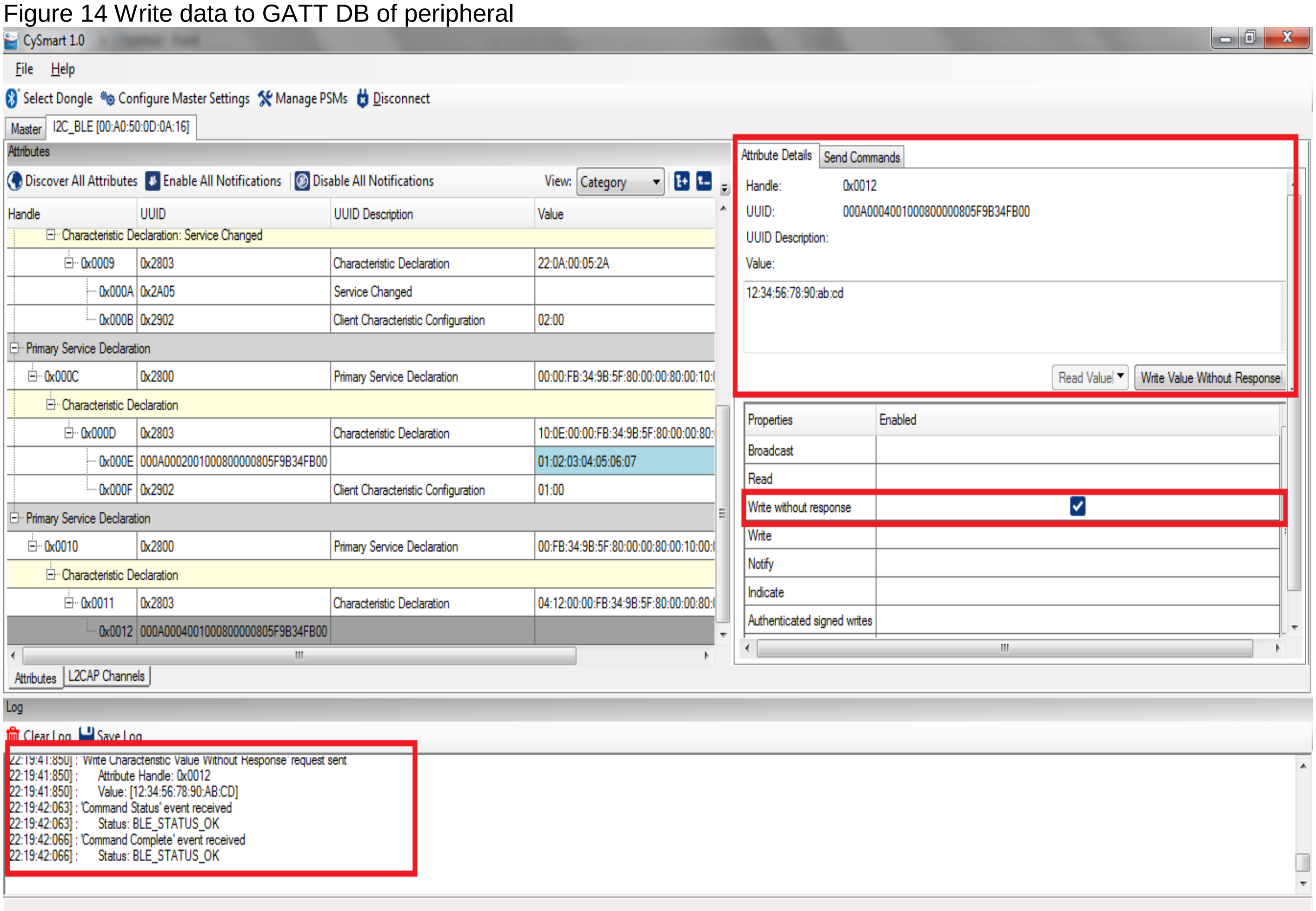
16. The data you wrote to the slave will be sent as notifications to the client.

Figure 13 Check notifications received



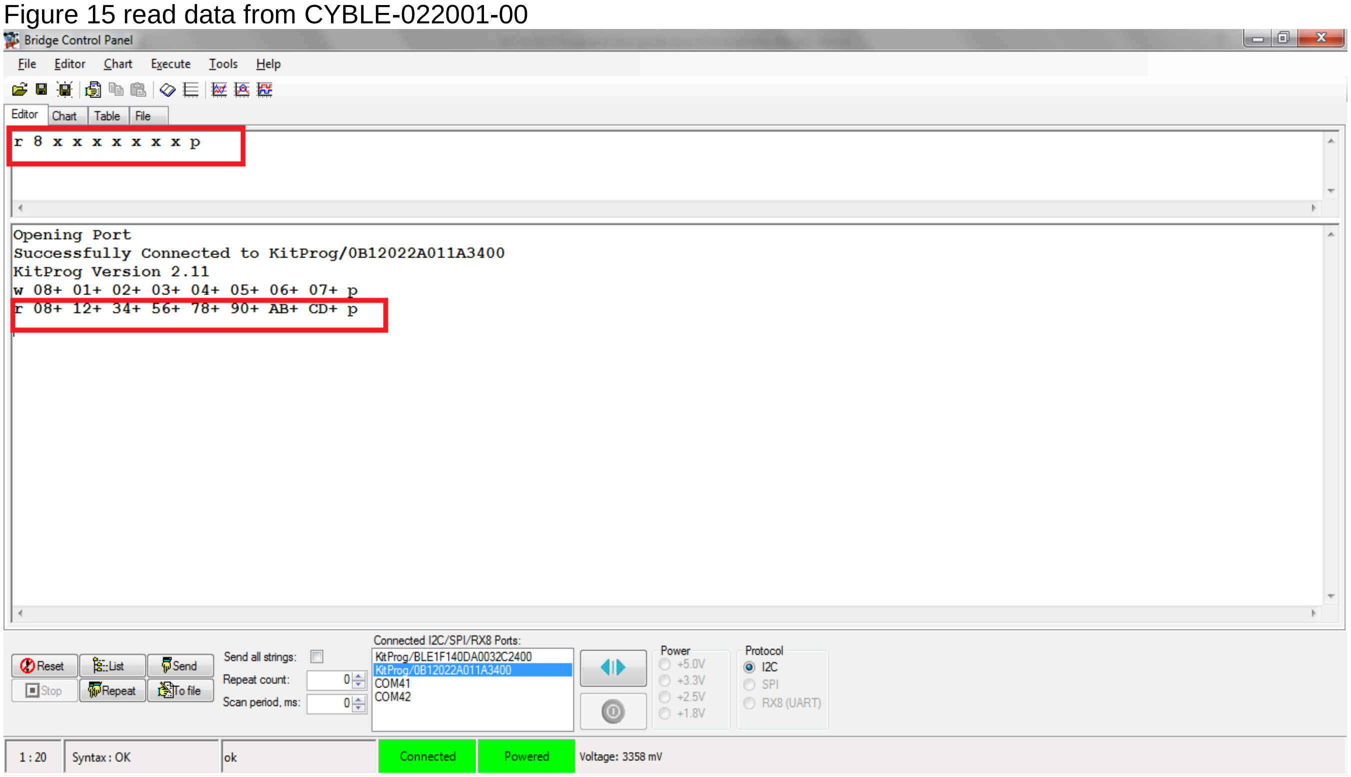
17. On CySmart select the handle 0x0012 and perform write without response to write to peripheral GATT DB

Figure 14 Write data to GATT DB of peripheral

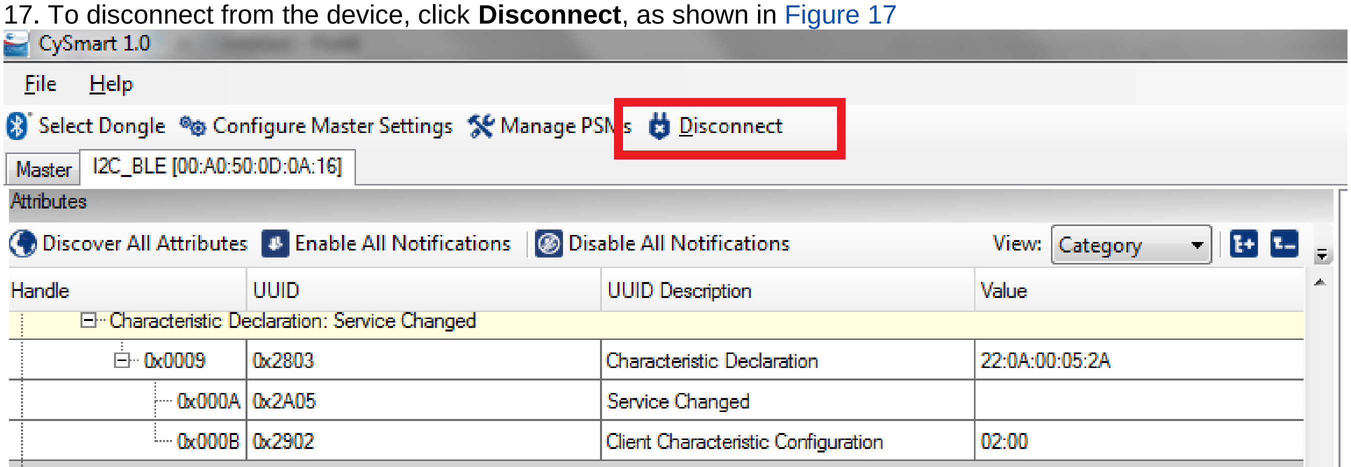


18. On Bridge control panel, read data from CYBLE-022001-00 by entering the command r 8 x1 x2 x3 x4 x5 x6 x7 p where r stands for read operation, 8 stands for clave address and x1 x2 x3 x4..xn stands for number of bytes to be read from I2C slave. n has to be less than 61.

Figure 15 read data from CYBLE-022001-00



17. To disconnect from the device, click **Disconnect**, as shown in Figure 17



18. After disconnection the RED LED will glow for 2 seconds and the device will start advertising as indicated by BLUE LED turning on. You can connect it back following the steps mentioned above.