

Objective

This example demonstrates how to use CYBLE-022001-00 BLE module as a BLE-UART bridge.

Overview:

This code example uses a custom BLE profile to demonstrate the BLE-UART bridge functionality. In order to emulate both peer devices as UART COM ports, this example essentially consists of two projects – Central and Peripheral. When appropriately interfaced to a PC, each of the peer devices can be used as a COM port using a terminal application (like TeraTerm). The data sent through the terminal application at one end, appears at the other end, as it does over wired UART communication.

Requirements:

<i>Programming Language</i>	: C (GCC 4.8.4)
<i>Associated Parts</i>	: CYBLE-022001-00 module
<i>Required software</i>	: PSoC Creator 3.1 , Terminal application (like TeraTerm)
<i>Required hardware</i>	: CYBLE-022001-00, CY8CKIT-042-BLE Bluetooth® Low Energy (BLE) Pioneer Kit
<i>Optional hardware</i>	: UART transceiver with flow control (if flow control is needed)

Project Description:

The BLE-UART project demonstrates the following:

- Custom service implementation
- BLE-UART functionality with an optional UART flow control
- Low power implementation for coin-cell operation

The custom service (named as “*Server_UART*”) used in this project has two characteristics: *Server_UART_Tx_data* and *Server_UART_Rx_data*. The *Server_UART_Tx_data* characteristic simulates the TX path of wired UART by sending ‘*notifications*’ from the Peripheral device to the Central device. While *Server_UART_Rx_data* characteristic simulates the RX path of wired UART by receiving ‘*write requests*’ from the Central device.

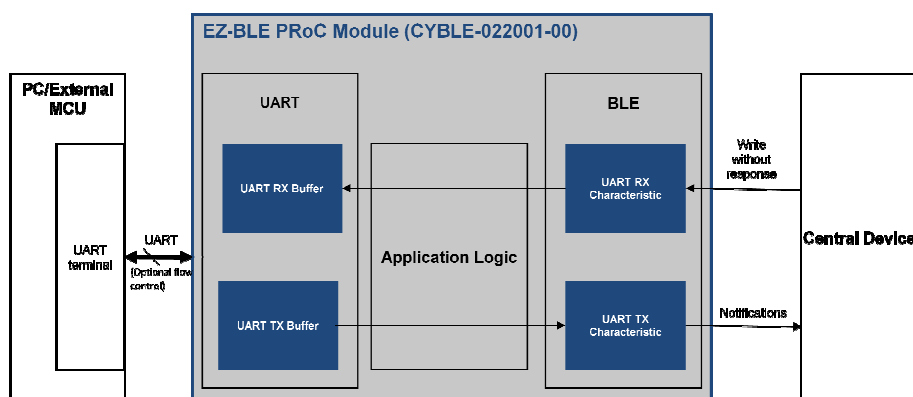


Figure 1: CYBLE-022001-00 Module configured as BLE-UART peripheral device

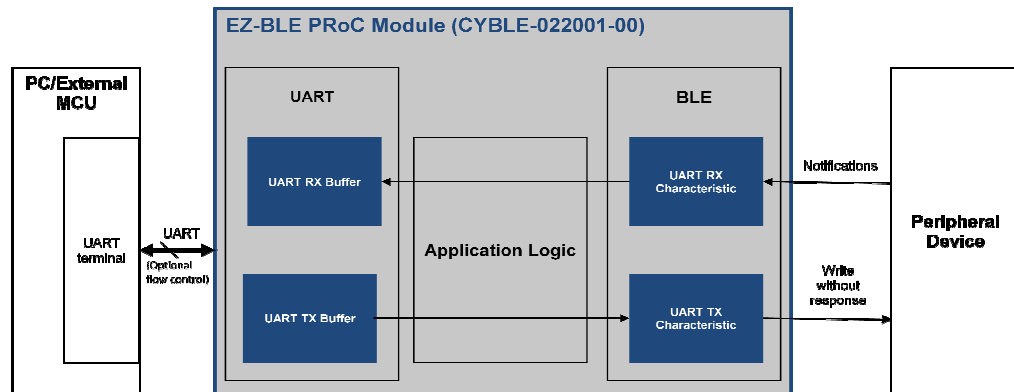


Figure 2: CYBLE-022001-00 Module configured as BLE-UART central device

These properties for the custom service/characteristics are configured in the BLE component of the "UART_to_BLE_peripheral_project", as shown in figure below:

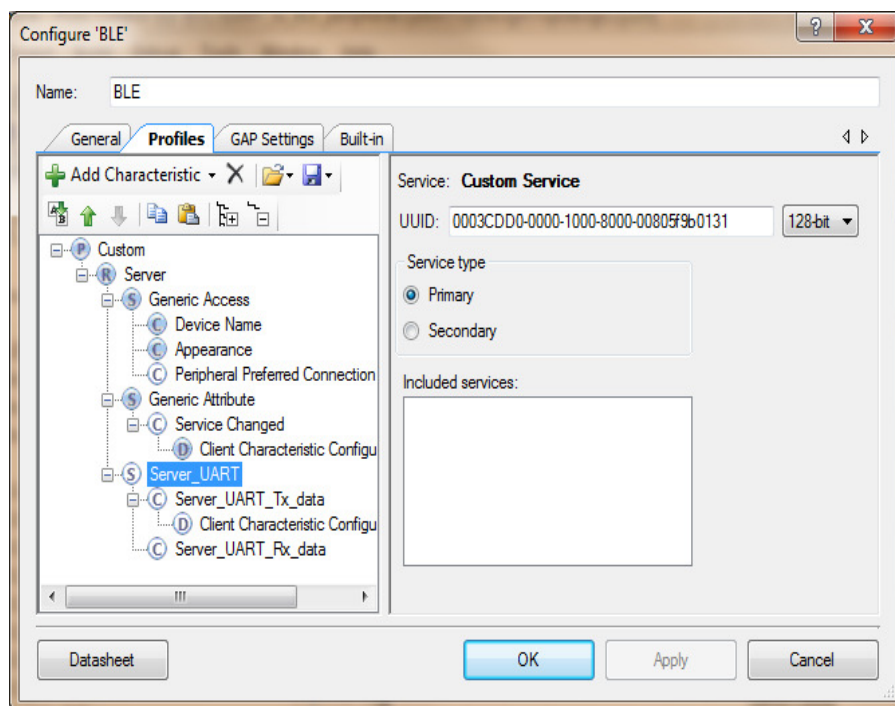


Figure 3: Custom service in BLE component configuration window

Hardware Setup:

Since BLE Pioneer Kit enumerates as a USB-UART, it can be directly be used for interfacing the module to a UART terminal application. Hence, the only hardware setup required is two pioneer kits connected to a PC through USB. In this setup, the project can work in its default configuration:

Baud rate : 115200bps
Flow Control : DISABLED
UART Rx Pin : P1.4 (this pin is fixed if Pioneer kit USB-UART is being used)
UART Tx Pin : P1.5 (this pin is fixed if Pioneer kit USB-UART is being used)

However, if UART flow control or a baud rate of greater than 115200 is to be used, an external UART transceiver should be used by appropriately connecting the UART pins. In order to enable Flow Control, the following changes are needed in the project:

1. Select the RTS and CTS check boxes in the flow control section of the UART configuration window.

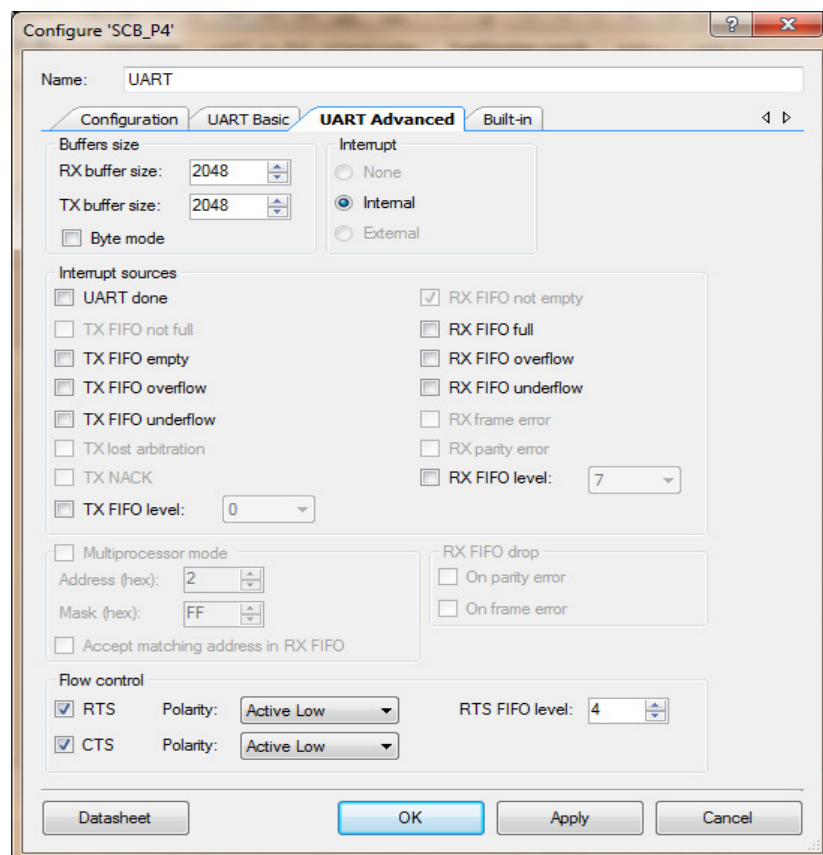


Figure 4: UART configuration window

2. Since pioneer kit USB-UART does not support flow control, P1.4 and P1.5 cannot be used as UART RX and TX pins. Select a different set of pins as UART pins for interfacing to external UART transceiver. An example of pin mapping for flow control is shown in below image:

Alias	Name /	Port
	\UART:cts\	P1[7] OA3:vplus_alt, TCPWM3:line_out_co SCB0:uart cts, SCB0:spi clk
	\UART:rts\	P1[6] OA2:vplus_alt, TCPWM3:line_out, SCB0:uart rts, SCB0:spi select[0]
	\UART:rx\	P0[4] LPCOMP:in_p[1], TCPWM1:line_out, SCB0:uart_rx, SRSS:ext_clk, SCB0:i2c_sda, SCB0:spi mosi
	\UART:tx\	P0[5] LPCOMP:in_n[1], TCPWM1:line_out_c SCB0:uart_tx, SCB0:i2c_scl, SCB0:spi miso
	Adv_LED	P3[6] SARMUX:pads[6], TCPWM3:line_out, SCB1:uart rts
	Conn_LED	P3[7] SARMUX:pads[7], TCPWM3:line_out_c SCB1:uart cts, SRSS:ext_clk lf

Figure 5: Pin configuration window

3. Uncomment the macro for flow control in "main.h".

```

/*****
 * Conditional compilation parameters
 *****/
#define FLOW_CONTROL
#define PRINT_MESSAGE_LOG
#define LOW_POWER_MODE

```

Figure 6: Macros in "main.h"

Building and Programming the device:

This section shows how to build the project and program CYBLE-022001-00 module. If you are using a development kit with a built-in programmer (BLE Pioneer Kit, for example), connect the BLE Pioneer Baseboard to your computer using the USB Standard-A to Mini-B cable. For other kits, refer to the kit user guide.

If you are developing on your own hardware, you need a hardware debugger, for example, a Cypress [CY8CKIT-002 MiniProg3](#).

1. Open the project "**UART_to_BLE_peripheral**" using PSoC Creator.
2. Select **Debug > Program** to program the device with the project, as shown in figure.

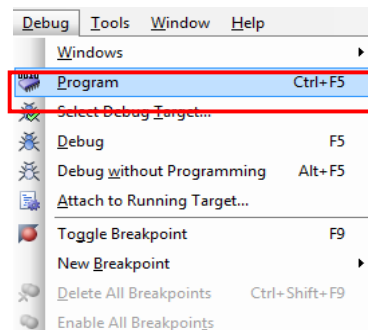


Figure 7: Programming device

3. The status bar (lower-left corner of the window) displays the programming status, as shown in figure.

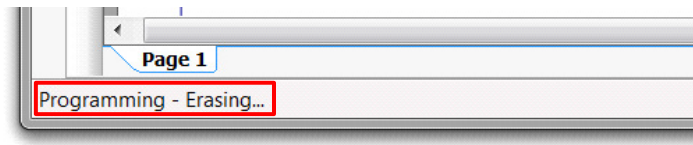


Figure 8: Programming status

4. Repeat steps (1) to (8) for the project “**UART_to_BLE_central**”, on a different module.

Testing and Expected Results:

1. After programming, plug the BLE Pioneer Kit to your computer’s USB port. As soon as the device gets powered
 - i. the RED LED should start blinking (for central role), indicating BLE SCANNING
 - ii. the GREEN LED should start blinking (for peripheral role), indicating BLE ADVERTISING
2. Open a PC application (like TeraTerm) and configure the desired COM port (in terms of baud rate, flow control, etc.) as per configuration used in the project.

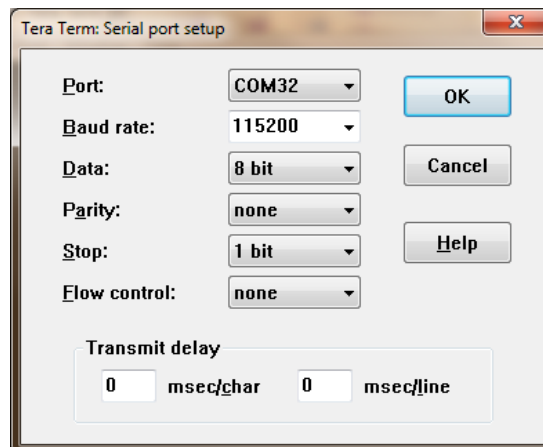


Figure 9: COM port configuration

3. If the macro “**PRINT_MESSAGE_LOG**” is uncommented (uncommented in default project) in “**main.h**” of the PSoC Creator project, the configuration information should appear as shown in figure.

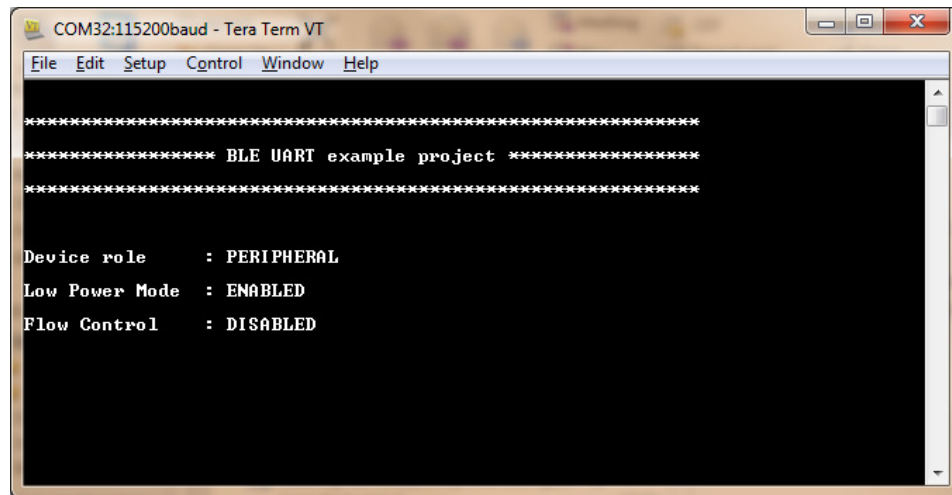


Figure 10: Project configuration information with *PRINT_MESSAGE_LOG* macro

4. If both the devices are in range of each other, they should automatically connect and BLUE LED should start blinking on both of them. The application corresponding to both should display messages, as shown in the following figures:

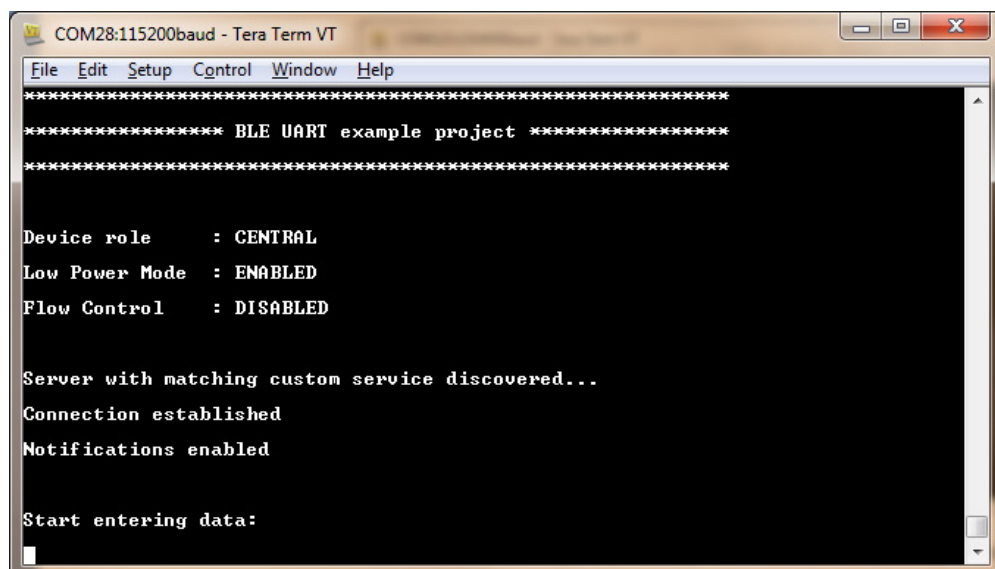
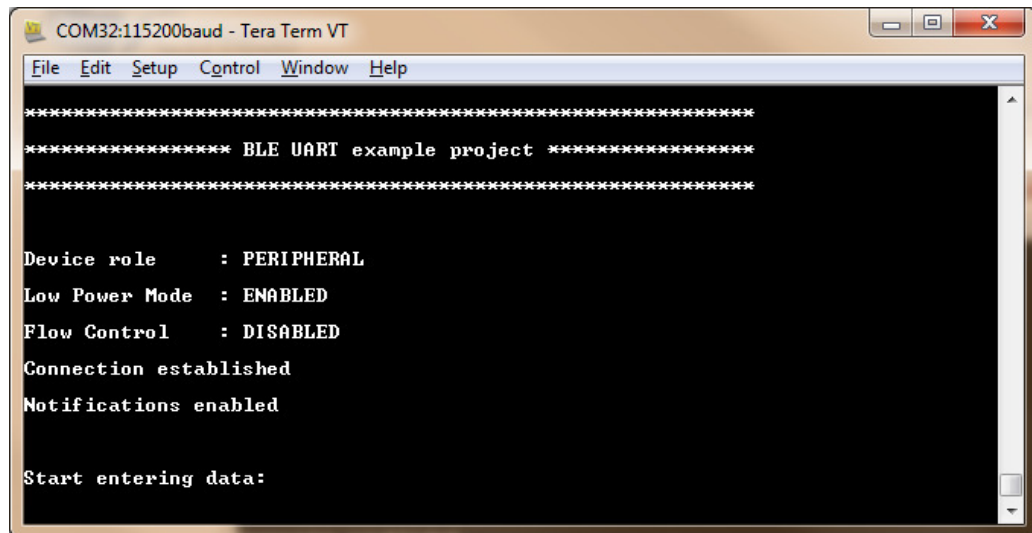


Figure 11: Message log for *UART_to_BLE_central* project

Figure 12: Message log for *UART_to_BLE_peripheral* project

5. The data entered on either side should appear on the other side.

Measuring the throughput:

1. For measuring the throughput, go to **File > Send file...**, browse for a .txt file and select open.

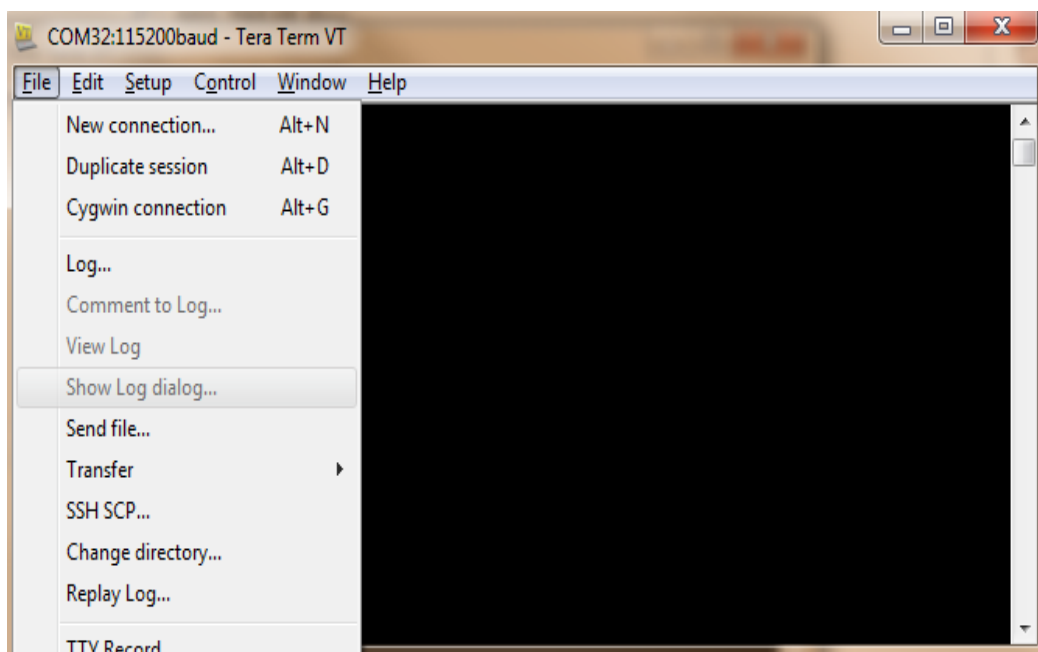


Figure 13: 'Send file...' option in TeraTerm

2. While the file is being sent, a dialog box (similar to below figure) should appear which displays the throughput.

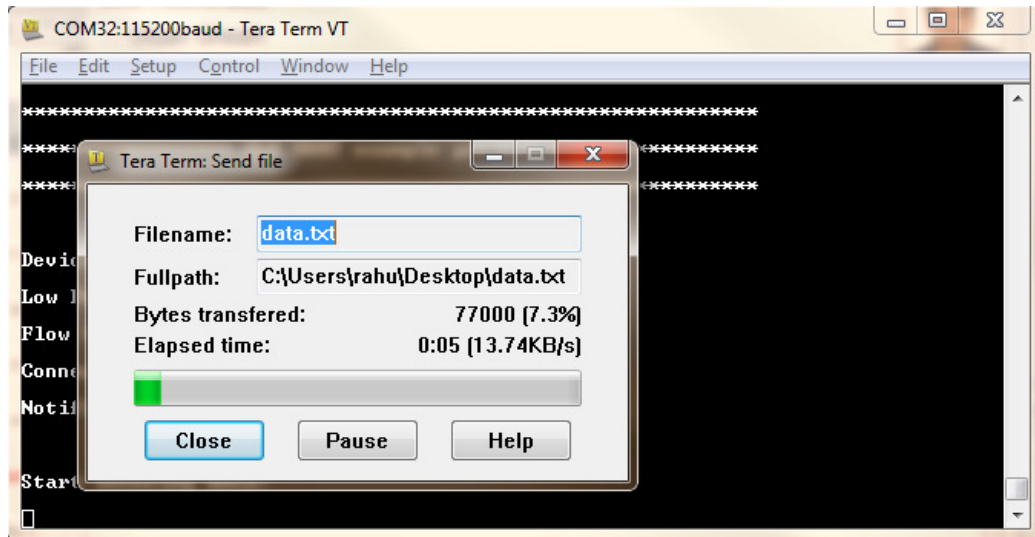


Figure 14: Throughput display in TeraTerm

3. If data integrity is to be verified, the TeraTerm log on the receiver end can be copied to a .txt file, which can be compared to the source .txt file using a compare tool (like Notepad++, etc.).

Related Documents

Table 1 lists all relevant application notes, code examples, knowledge base articles, device datasheets, and Component / user module datasheets.

Table 1: Related Documents

Document	Title	Comment
AN91267	Getting Started with PSoC4 BLE	Provides an introduction to PSoC4 BLE device that integrates a Bluetooth Low Energy radio system along with programmable analog and digital resources.