CYBLE-022001-00 – UART-BLE

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:

Programming Language	**:** C (GCC 4.8.4)
Associated Parts	**:** CYBLE-022001-00 module
Required software	**:** PSoC Creator 3.1, Terminal application (like TeraTerm)
Required hardware	**:** CYBLE-022001-00, CY8CKIT-042-BLE Bluetooth® Low Energy (BLE) Pioneer Kit
Optional hardware	**:** 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.

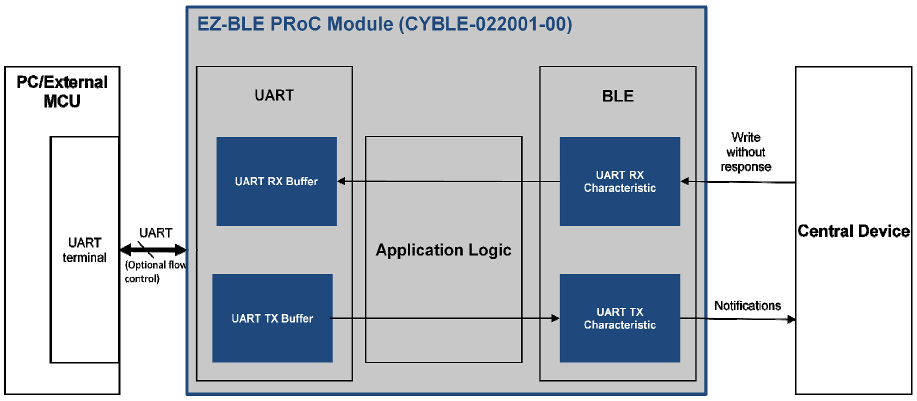


Figure 1: CYBLE-022001-00 Module configured as BLE-UART peripheral device

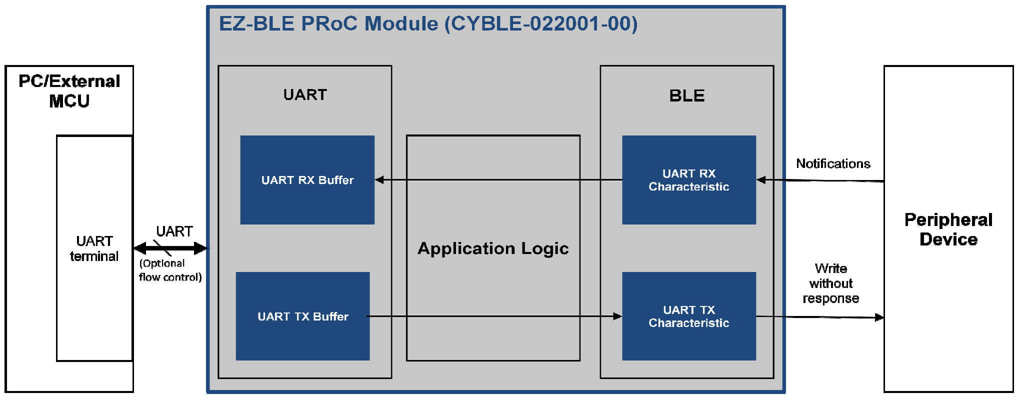
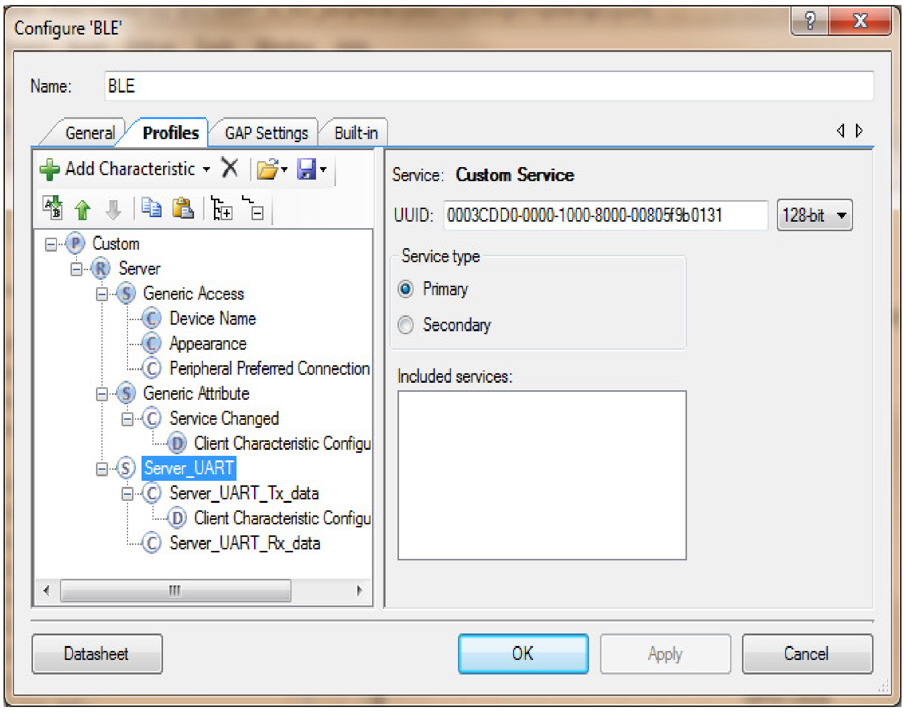


Figure 2: CYBLE-022001-00 Module configured as BLE-UART central device

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:



Name: BLE

- Custom
  - Server
    - Generic Access
      - Device Name
      - Appearance
      - Peripheral Preferred Connection
    - Generic Attribute
      - Service Changed
        - Client Characteristic Configu
    - Server_UART
      - Server_UART_Tx_data
        - Client Characteristic Configu
      - Server_UART_Rx_data

Service: **Custom Service**

UUID: 0003CDD0-0000-1000-8000-00805f9b0131 128-bit

Service type

- [✓ Primary]
- Secondary

Included services:

Figure 3: Custom service in BLE component configuration window

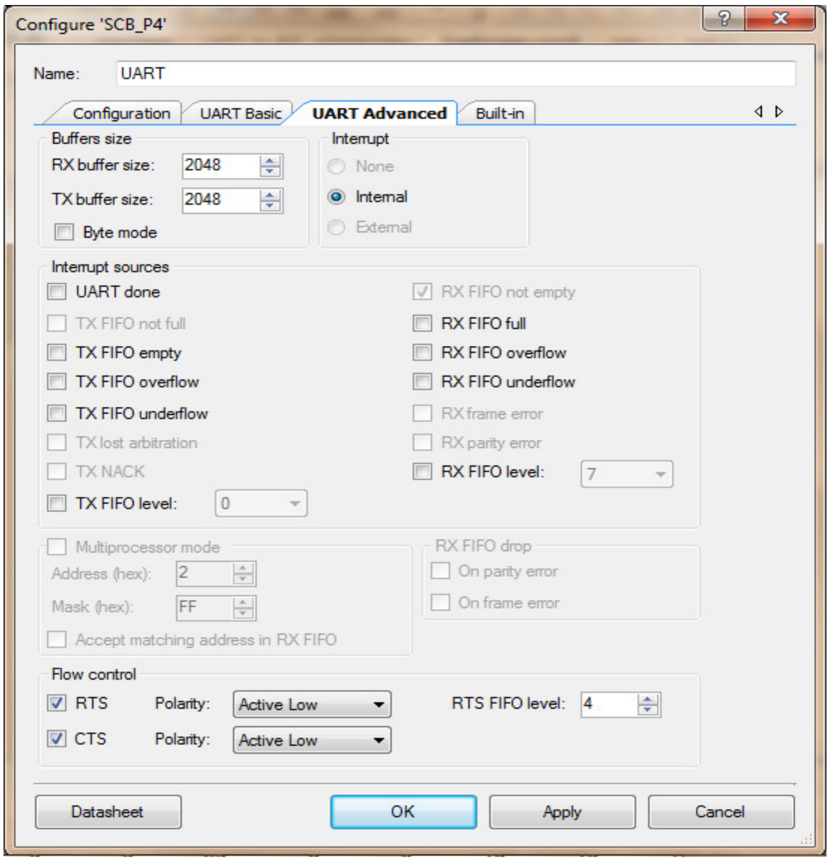
Hardware Setup:

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.



Configure 'SCB_P4'

Name: UART

Configuration | UART Basic | **UART Advanced** | Built-in

Buffers size
- RX buffer size: 2048
- TX buffer size: 2048
- [ ] Byte mode

Interrupt
- ( ) None
- (●) Internal
- ( ) External

Interrupt sources
- [ ] UART done
- [ ] TX FIFO not full
- [ ] TX FIFO empty
- [ ] TX FIFO overflow
- [ ] TX FIFO underflow
- [ ] TX lost arbitration
- [ ] TX NACK
- [ ] TX FIFO level: 0
- [✓] RX FIFO not empty
- [ ] RX FIFO full
- [ ] RX FIFO overflow
- [ ] RX FIFO underflow
- [ ] RX frame error
- [ ] RX parity error
- [ ] RX FIFO level: 7

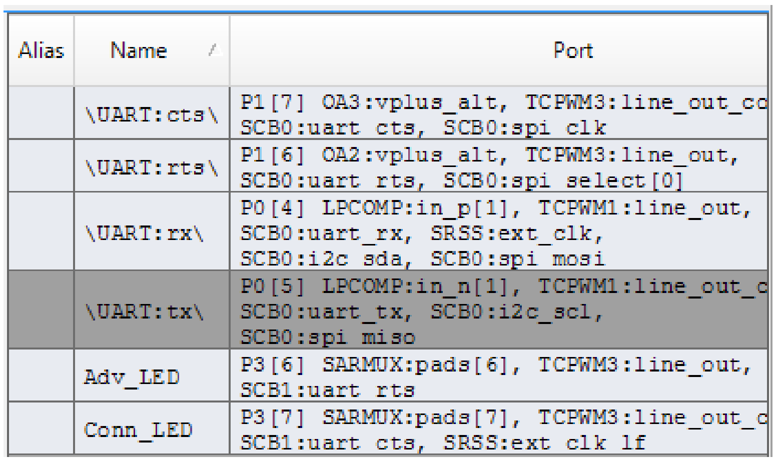
- [ ] Multiprocessor mode
- Address (hex): 2
- Mask (hex): FF
- [ ] Accept matching address in RX FIFO

RX FIFO drop
- [ ] On parity error
- [ ] On frame error

Flow control
- [✓] RTS Polarity: Active Low RTS FIFO level: 4
- [✓] CTS Polarity: Active Low

Figure 4: UART configuration window

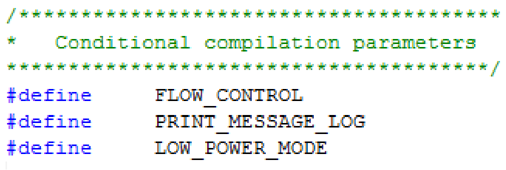
2. Since pioneer kit USB-UART does not support flow control, P1.4 and P1.5 cannot be used as UART RX and TX pins. Select a different set of pins as UART pins for interfacing to external UART transceiver. An example of pin mapping for flow control is shown in below image:



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.

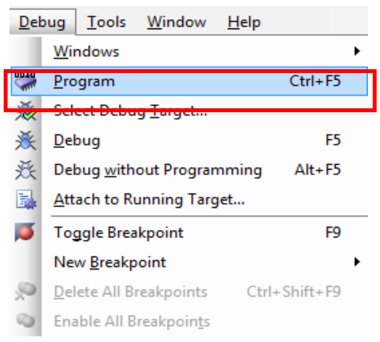
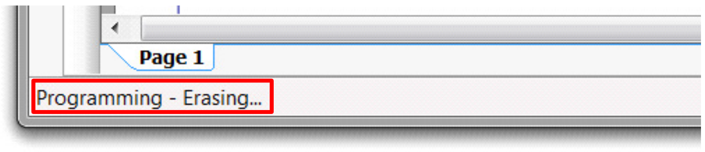


Figure 7: Programming device

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



Page 1

Programming - Erasing...

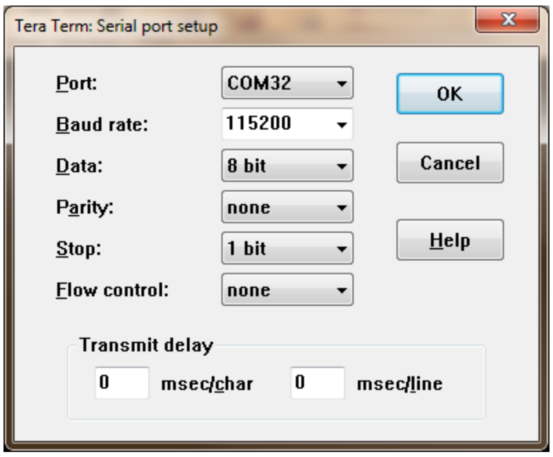
Figure 8: Programming status

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.



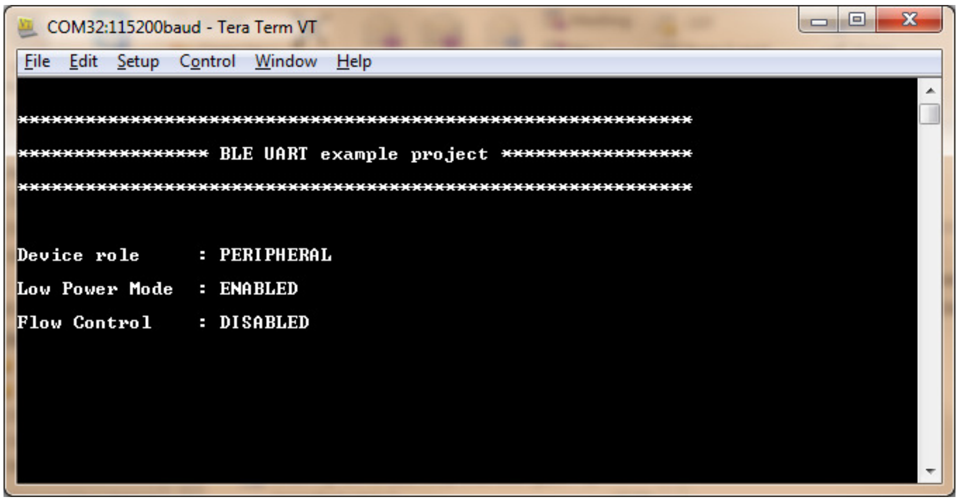
Tera Term: Serial port setup

Port: COM32
Baud rate: 115200
Data: 8 bit
Parity: none
Stop: 1 bit
Flow control: none

Transmit delay
0 msec/char
0 msec/line

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.

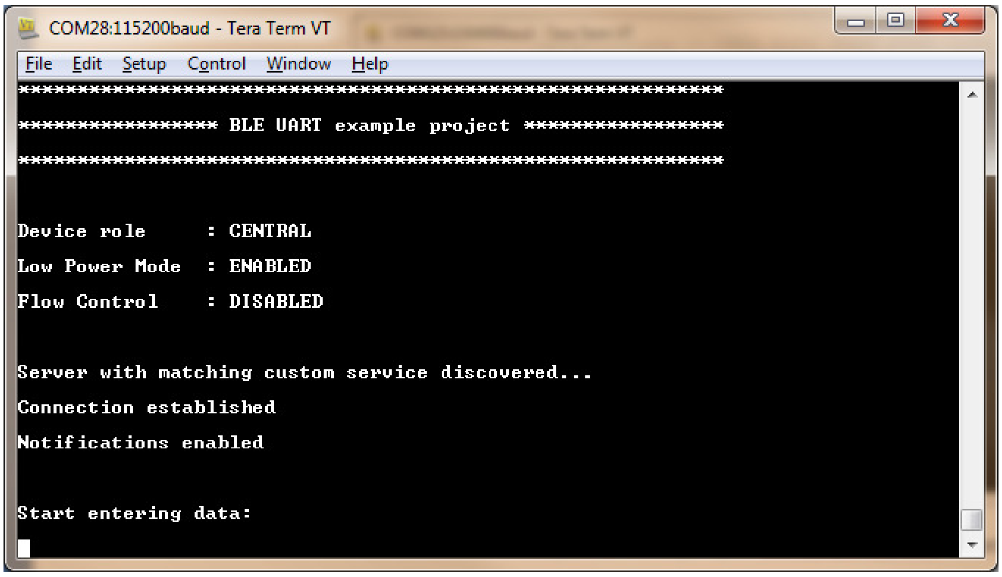

COM32:115200baud - Tera Term VT
File Edit Setup Control Window Help

```
************************************************************
***************** BLE UART example project *****************
************************************************************

Device role      : PERIPHERAL
Low Power Mode   : ENABLED
Flow Control     : DISABLED
```

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:


COM28:115200baud - Tera Term VT
File Edit Setup Control Window Help

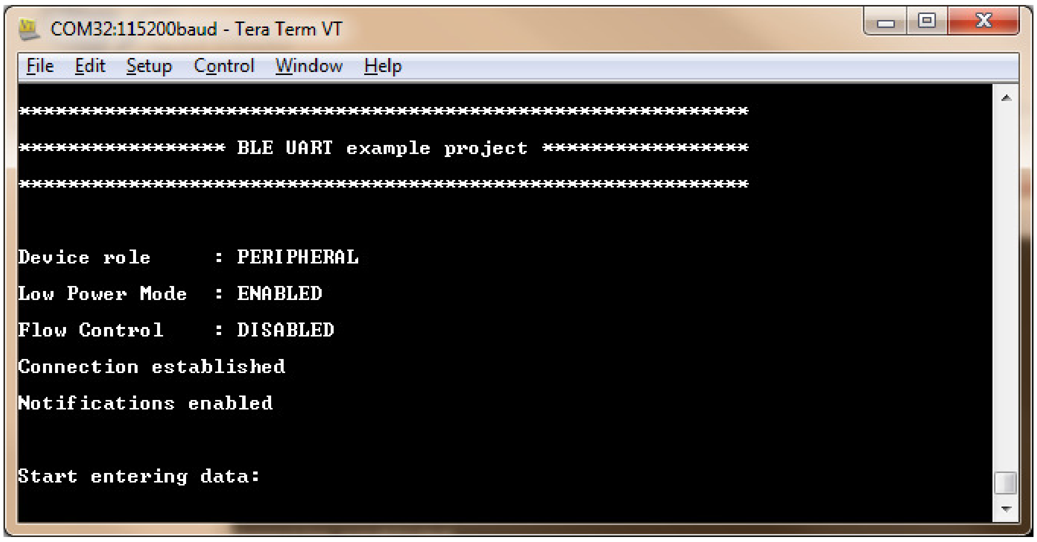
```
************************************************************
***************** BLE UART example project *****************
************************************************************

Device role      : CENTRAL
Low Power Mode   : ENABLED
Flow Control     : DISABLED

Server with matching custom service discovered...
Connection established
Notifications enabled

Start entering data:
```

Figure 11: Message log for *UART_to_BLE_central* project



```
COM32:115200baud - Tera Term VT
File  Edit  Setup  Control  Window  Help

************************************************************
***************** BLE UART example project *****************
************************************************************

Device role      : PERIPHERAL
Low Power Mode   : ENABLED
Flow Control     : DISABLED
Connection established
Notifications enabled

Start entering data:
```

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.

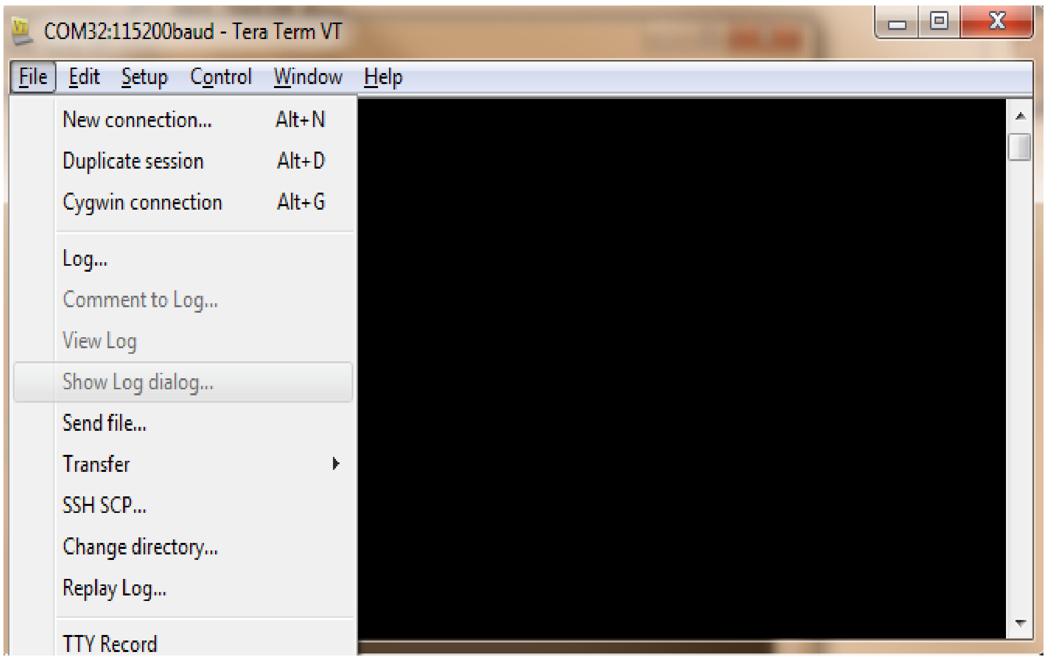
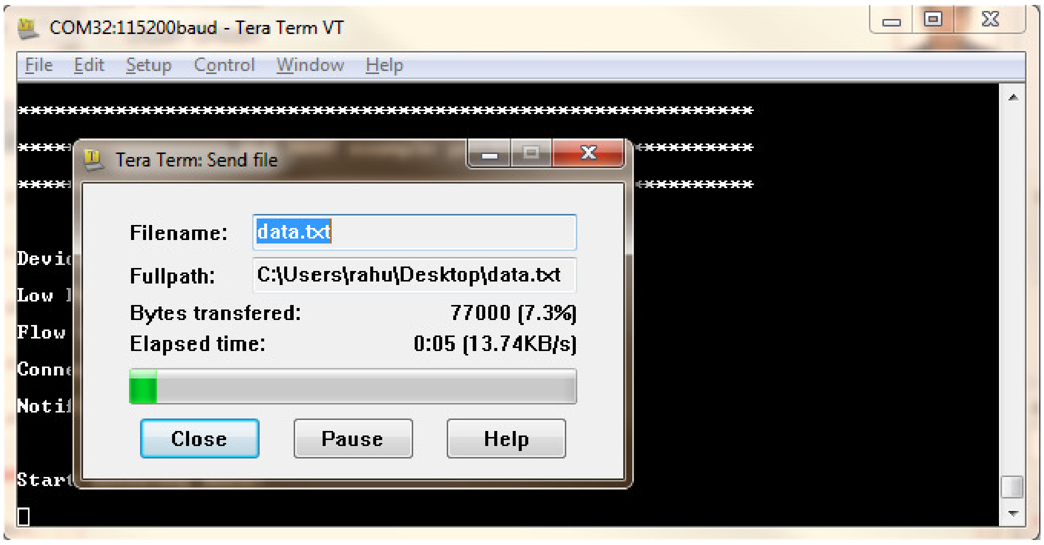


Figure 13: ‘Send file…’ option in TeraTerm

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



COM32:115200baud - Tera Term VT

Tera Term: Send file

Filename: data.txt
Fullpath: C:\Users\rahu\Desktop\data.txt
Bytes transfered: 77000 (7.3%)
Elapsed time: 0:05 (13.74KB/s)

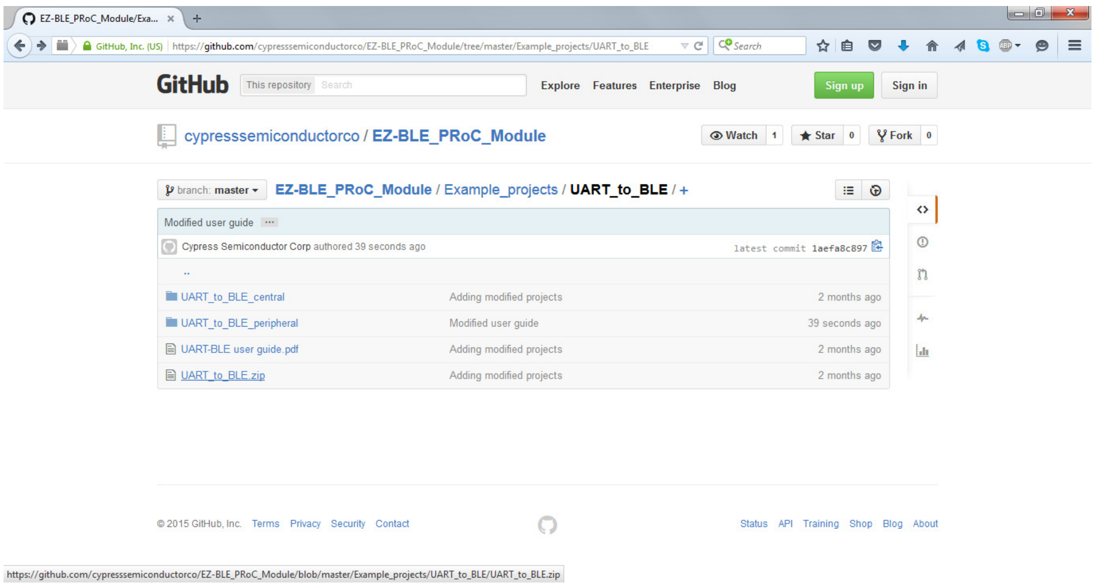
Figure 14: Throughput display in TeraTerm

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.).
4.

Downloading project from GitHub:

The following steps should be followed to download the complete set of projects (both Central and Peripheral) and user guide:

1. Go to the project page on GitHub.com and select “UART_to_BLE.zip” as shown in the figure below.



https://github.com/cypresssemiconductorco/EZ-BLE_PRoC_Module/tree/master/Example_projects/UART_to_BLE

cypresssemiconductorco / EZ-BLE_PRoC_Module

branch: master — EZ-BLE_PRoC_Module / Example_projects / UART_to_BLE / +

Modified user guide

Cypress Semiconductor Corp authored 39 seconds ago — latest commit 1aefa8c897

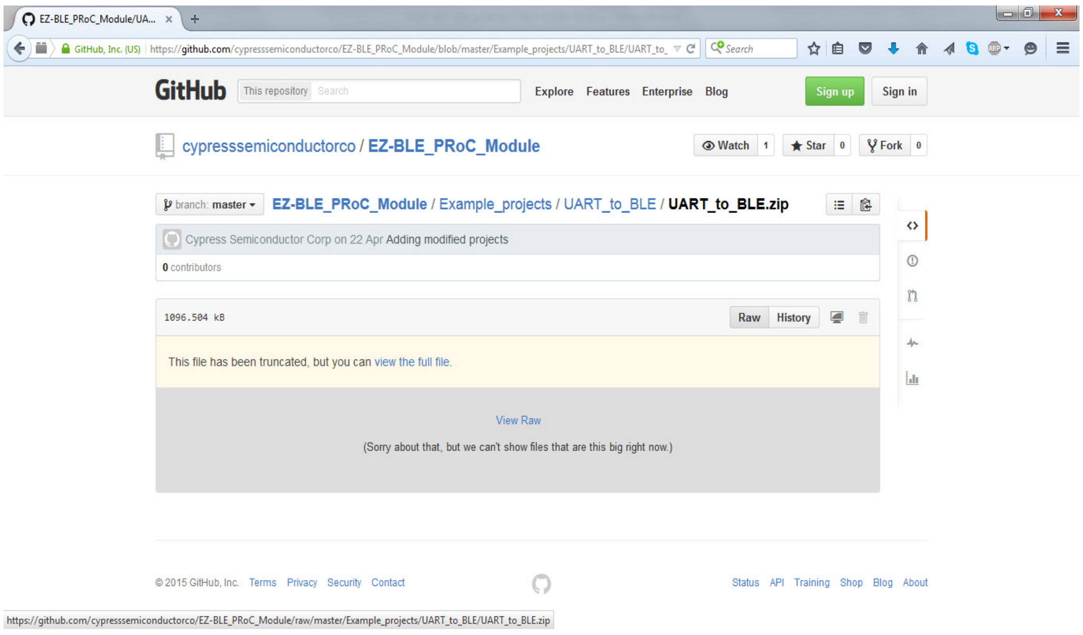
UART_to_BLE_central	Adding modified projects	2 months ago
UART_to_BLE_peripheral	Modified user guide	39 seconds ago
UART-BLE user guide.pdf	Adding modified projects	2 months ago
UART_to_BLE.zip	Adding modified projects	2 months ago

© 2015 GitHub, Inc.

https://github.com/cypresssemiconductorco/EZ-BLE_PRoC_Module/blob/master/Example_projects/UART_to_BLE/UART_to_BLE.zip

Figure 15: Selecting “UART_to_BLE.zip”

2. Select “Raw” button, as shown in the following figure. All the project files and user guide will be downloaded as a .zip file.


EZ-BLE_PRoC_Module/UA...

https://github.com/cypresssemiconductorco/EZ-BLE_PRoC_Module/blob/master/Example_projects/UART_to_BLE/UART_to_

GitHub This repository Search Explore Features Enterprise Blog Sign up Sign in

cypresssemiconductorco / EZ-BLE_PRoC_Module Watch 1 Star 0 Fork 0

branch: master EZ-BLE_PRoC_Module / Example_projects / UART_to_BLE / **UART_to_BLE.zip**

Cypress Semiconductor Corp on 22 Apr Adding modified projects

0 contributors

1096.504 kB Raw History

This file has been truncated, but you can view the full file.

View Raw

(Sorry about that, but we can't show files that are this big right now.)

© 2015 GitHub, Inc. Terms Privacy Security Contact Status API Training Shop Blog About

https://github.com/cypresssemiconductorco/EZ-BLE_PRoC_Module/raw/master/Example_projects/UART_to_BLE/UART_to_BLE.zip

Figure 16: Downloading “UART_to_BLE.zip”

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.