UART HCI Bluetooth Module for Linux

BT860

Application Note *v1.0*

Introduction

BT860 is Laird's latest UART HCI Bluetooth module based on the Cypress CYW20704 A2 chipset. This application note describes how to use the BlueZ BCCMD tool to attach on the Linux platform. BlueZ and BlueZ-Utils packages are required for this operation.

Requirements

- BT860 development board
- BlueZ – Official Linux Bluetooth protocol stack

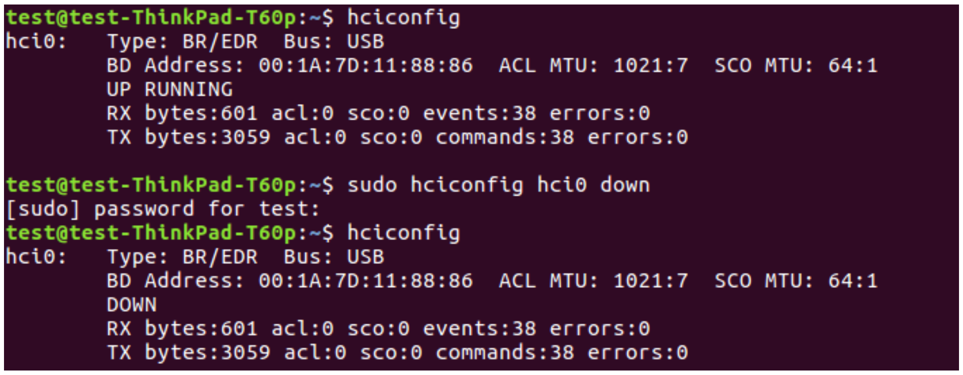
Notes: The BT860 development board uses the FTDI USB-UART chip. The testing platform used in this application note has the driver installed automatically. The name of the serial port is **/dev/ttyUSB0**.

Ubuntu 16.04 is used as the testing platform (Kernel version 4.4.0-31). The BlueZ stack (v 5.37) is included.

Preparation

Before plugging the BT860 development board to the computer, type ***hciconfig*** to find out if there are any existing Bluetooth radios. If you find one, close it by typing the following: ***hciconfig hciX down*** (Figure 1).

Typically, *hci0* is the first Bluetooth device on the computer. Superuser permissions should be required.

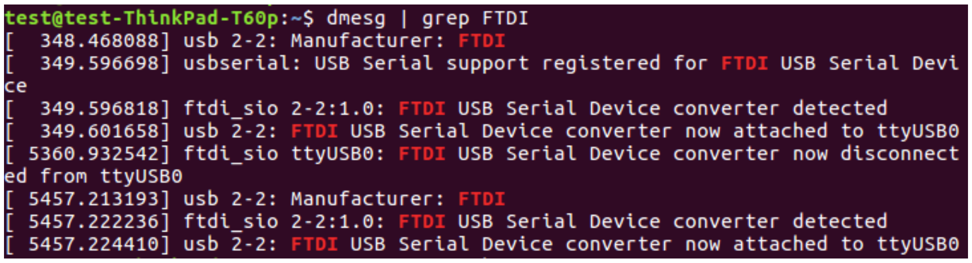


```
test@test-ThinkPad-T60p:~$ hciconfig
hci0:   Type: BR/EDR  Bus: USB
        BD Address: 00:1A:7D:11:88:86  ACL MTU: 1021:7  SCO MTU: 64:1
        UP RUNNING
        RX bytes:601 acl:0 sco:0 events:38 errors:0
        TX bytes:3059 acl:0 sco:0 commands:38 errors:0

test@test-ThinkPad-T60p:~$ sudo hciconfig hci0 down
[sudo] password for test:
test@test-ThinkPad-T60p:~$ hciconfig
hci0:   Type: BR/EDR  Bus: USB
        BD Address: 00:1A:7D:11:88:86  ACL MTU: 1021:7  SCO MTU: 64:1
        DOWN
        RX bytes:601 acl:0 sco:0 events:38 errors:0
        TX bytes:3059 acl:0 sco:0 commands:38 errors:0
```

Figure 1: Disable existing computer existing Bluetooth device

After plugging the BT860 development board to the computer, locate the USB UART port by typing the following: ***dmesg | grep FTDI*** (Figure 2). The development board uses the FTDI USB-UART chip.



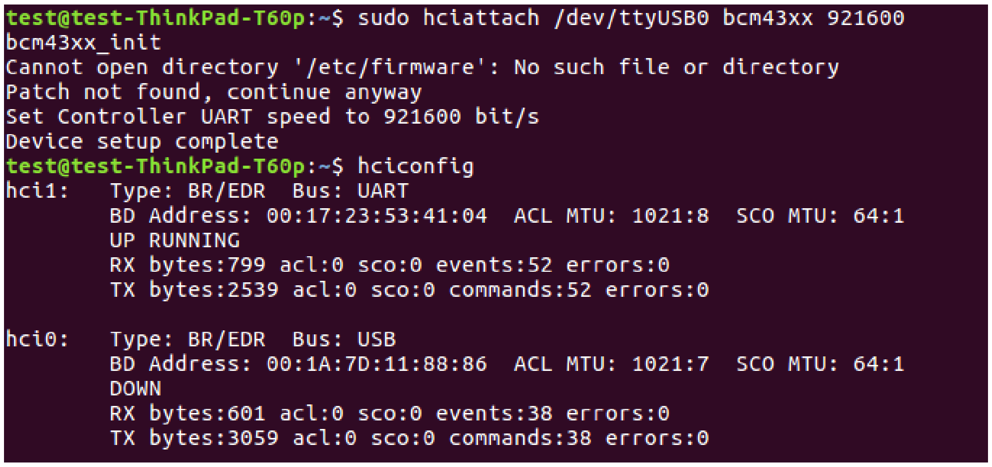
```
test@test-ThinkPad-T60p:~$ dmesg | grep FTDI
[  348.468088] usb 2-2: Manufacturer: FTDI
[  349.596698] usbserial: USB Serial support registered for FTDI USB Serial Devi
ce
[  349.596818] ftdi_sio 2-2:1.0: FTDI USB Serial Device converter detected
[  349.601658] usb 2-2: FTDI USB Serial Device converter now attached to ttyUSB0
[ 5360.932542] ftdi_sio ttyUSB0: FTDI USB Serial Device converter now disconnect
ed from ttyUSB0
[ 5457.213193] usb 2-2: Manufacturer: FTDI
[ 5457.222236] ftdi_sio 2-2:1.0: FTDI USB Serial Device converter detected
[ 5457.224410] usb 2-2: FTDI USB Serial Device converter now attached to ttyUSB0
```

Figure 2: Locate the FTDI USB-UART port

Attaching the HCI UART BT Module

In the recent release of BlueZ, the *hciattach* command was deprecated and is replaced with *btattach*. On this test platform, both commands are supported. Your platform may have removed the *hciattach* command. This document covers both commands.

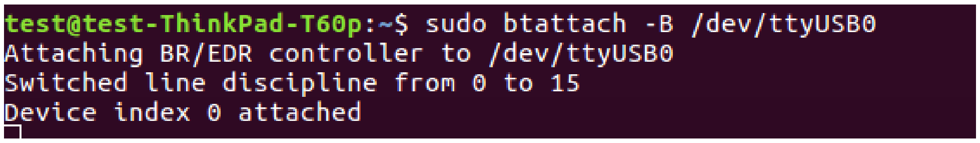
With the hciattach command, BlueZ tries to load the new firmware if it is provided. Even it is not provided, it continues to attach (Figure 3). The BT860 is loaded with HCI firmware at production (Figure 4).



```
test@test-ThinkPad-T60p:~$ sudo hciattach /dev/ttyUSB0 bcm43xx 921600
bcm43xx_init
Cannot open directory '/etc/firmware': No such file or directory
Patch not found, continue anyway
Set Controller UART speed to 921600 bit/s
Device setup complete
test@test-ThinkPad-T60p:~$ hciconfig
hci1:   Type: BR/EDR  Bus: UART
        BD Address: 00:17:23:53:41:04  ACL MTU: 1021:8  SCO MTU: 64:1
        UP RUNNING
        RX bytes:799 acl:0 sco:0 events:52 errors:0
        TX bytes:2539 acl:0 sco:0 commands:52 errors:0

hci0:   Type: BR/EDR  Bus: USB
        BD Address: 00:1A:7D:11:88:86  ACL MTU: 1021:7  SCO MTU: 64:1
        DOWN
        RX bytes:601 acl:0 sco:0 events:38 errors:0
        TX bytes:3059 acl:0 sco:0 commands:38 errors:0
```

Figure 3: hciattach /dev/ttyUSB0 bcm43xx 921600 attaches the BT860

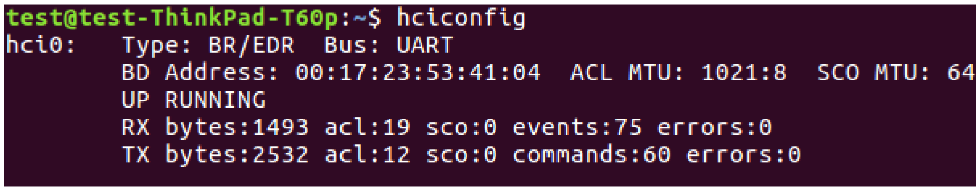


```
test@test-ThinkPad-T60p:~$ sudo btattach -B /dev/ttyUSB0
Attaching BR/EDR controller to /dev/ttyUSB0
Switched line discipline from 0 to 15
Device index 0 attached
```

Figure 4: btattach /dev/ttyUSB0 attaches the BT860

Launching the Bluetooth Stack with New Settings

To confirm that the BT860 is successfully attached, type ***hciconfig*** to see all recognized Bluetooth radios. To enable the BT860, type ***hciconfig hci0 up*** if it is shown as DOWN (Figure 5).



```
test@test-ThinkPad-T60p:~$ hciconfig
hci0:   Type: BR/EDR  Bus: UART
        BD Address: 00:17:23:53:41:04  ACL MTU: 1021:8  SCO MTU: 64
        UP RUNNING
        RX bytes:1493 acl:19 sco:0 events:75 errors:0
        TX bytes:2532 acl:12 sco:0 commands:60 errors:0
```

Figure 5: BT860 is recognized and listed as UP and RUNNING

CONNECTING THE BT860 VIA COMMAND LINE

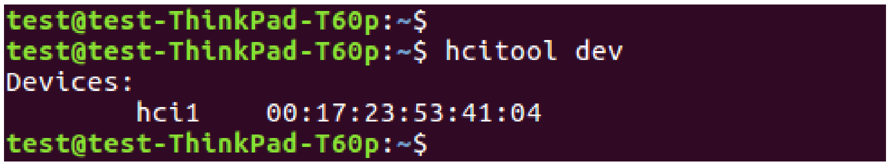
Verify BT860 Connection

In Linux, you may configure and test the BT860 via terminal. The Linux utility to configure and identify Bluetooth is *hcitool*. To verify that the BT860 is recognized by the operating system, run *hcitool* and check for devices by doing the following:

1. Open the command terminal.
2. Enter the following command:

```
hcitool dev
```

This command displays local devices. If it finds one, it returns the following (Figure 6):



```
test@test-ThinkPad-T60p:~$
test@test-ThinkPad-T60p:~$ hcitool dev
Devices:
        hci1    00:17:23:53:41:04
test@test-ThinkPad-T60p:~$
```

Figure 6: Command found a local device

Note: The hcitool command uses the first available Bluetooth device for its operations. If multiple Bluetooth devices are found, all hcitool commands must specify which device to use, as follows:

```
hcitool [-i <hciX>] [command [command parameters]]
```

In this example, <hciX> must correspond to the HCI device number found using hcitool dev, e.g. hci1.

Connecting with Classic Bluetooth

With the device initialized, you may test Bluetooth functionality from the command prompt. To test scanning, you must have a nearby device (such as a tablet or smartphone) set to be discoverable.

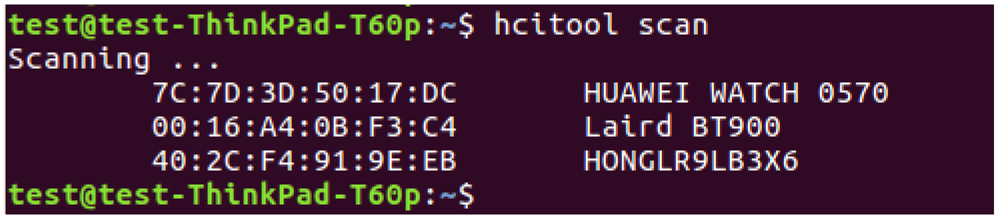
The command to initialize a scan is:

```
hcitool scan
```

When a scan is initialized, the terminal returns found devices in the following format:

```
Scanning ...
          [MAC Address]   Friendly_Name
```

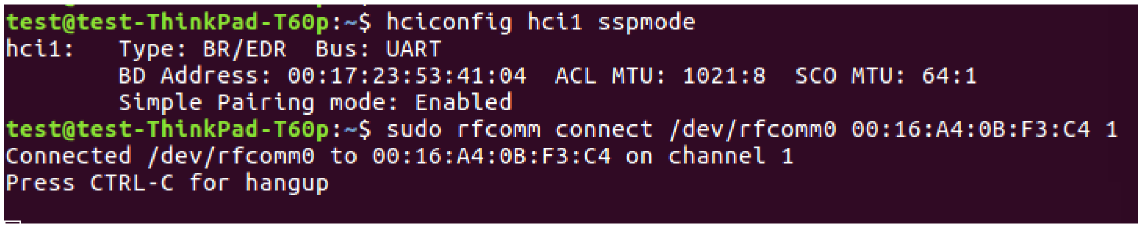
If there are discoverable devices nearby, they appear in this list as they are discovered (Figure 7).



```
test@test-ThinkPad-T60p:~$ hcitool scan
Scanning ...
        7C:7D:3D:50:17:DC       HUAWEI WATCH 0570
        00:16:A4:0B:F3:C4       Laird BT900
        40:2C:F4:91:9E:EB       HONGLR9LB3X6
test@test-ThinkPad-T60p:~$
```

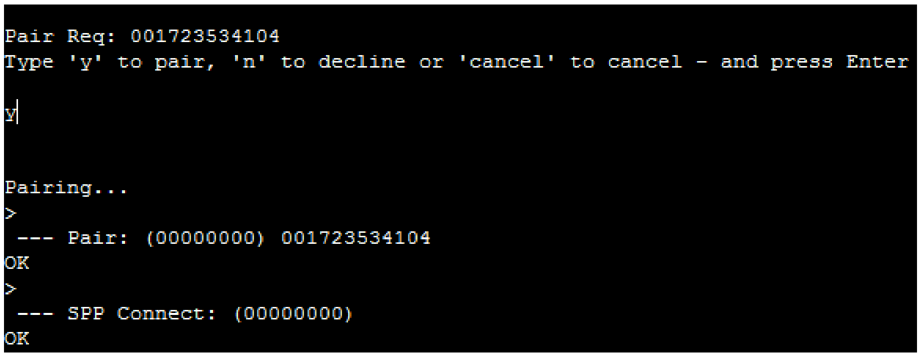
Figure 7: Laird module is discovered

To demonstrate the RFcomm connection, a Laird module (already configured as discoverable and connectable), is used. Simple secure mode must also be enabled (Figure 8).



```
test@test-ThinkPad-T60p:~$ hciconfig hci1 sspmode
hci1:   Type: BR/EDR  Bus: UART
        BD Address: 00:17:23:53:41:04  ACL MTU: 1021:8  SCO MTU: 64:1
        Simple Pairing mode: Enabled
test@test-ThinkPad-T60p:~$ sudo rfcomm connect /dev/rfcomm0 00:16:A4:0B:F3:C4 1
Connected /dev/rfcomm0 to 00:16:A4:0B:F3:C4 on channel 1
Press CTRL-C for hangup
```

Figure 8: RFconn connection to the module



```
Pair Req: 001723534104
Type 'y' to pair, 'n' to decline or 'cancel' to cancel - and press Enter

y


Pairing...
>
 --- Pair: (00000000) 001723534104
OK
>
 --- SPP Connect: (00000000)
OK
```

Figure 9: Module shows it is connected

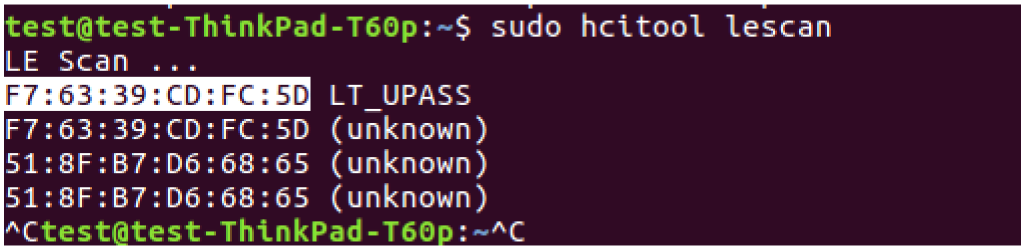
Connecting with Bluetooth Low Energy

The *hcitool* commands to scan Bluetooth Low Energy are distinct from those used in classic Bluetooth connections. To initiate a BLE scan from the terminal, issue the following command:

```
#hcitool lescan
```

The terminal returns the following:

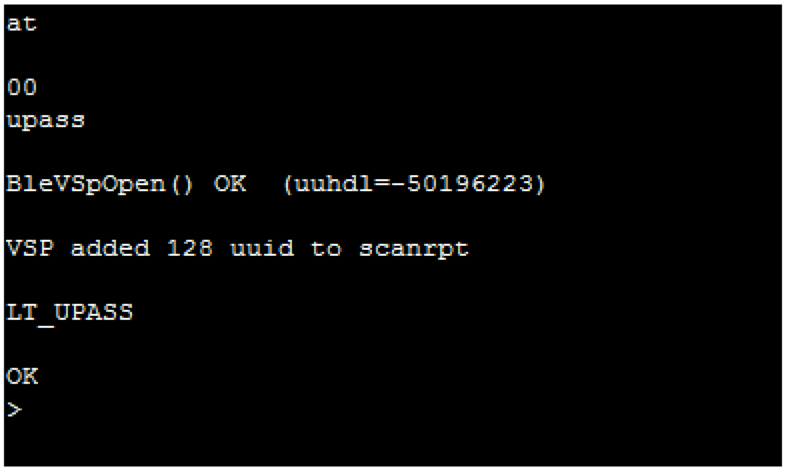
```
LE Scan ...
[MAC Address] – [BLE device]
```



```
test@test-ThinkPad-T60p:~$ sudo hcitool lescan
LE Scan ...
F7:63:39:CD:FC:5D LT_UPASS
F7:63:39:CD:FC:5D (unknown)
51:8F:B7:D6:68:65 (unknown)
51:8F:B7:D6:68:65 (unknown)
^Ctest@test-ThinkPad-T60p:~^C
```

Figure 10: Scan for BLE devices

To demonstrate the BLE connection, a Laird module running the Laird vSP upass application is used (Figure 11).



```
at

00
upass

BleVSpOpen() OK  (uuhdl=-50196223)

VSP added 128 uuid to scanrpt

LT_UPASS

OK
>
```

Figure 11: Module running $autorun$.vsp.UART.bridge application

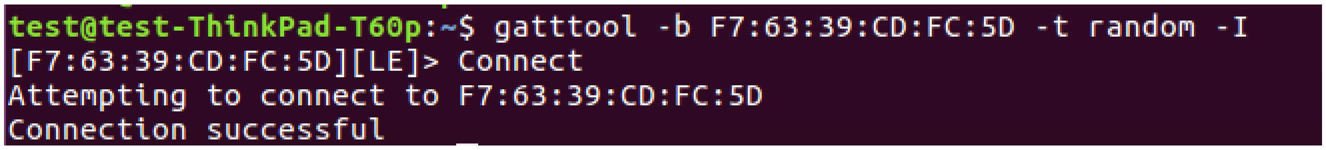
The Linux computer first scans for the Bluetooth device. Once the module displays, Press **Ctrl-C** to stop the scanning. Send the following command:

```
gatttool -b <BT900_MAC> -t random -I
```

Once the prompt is returned, send the following:

```
Connect
```

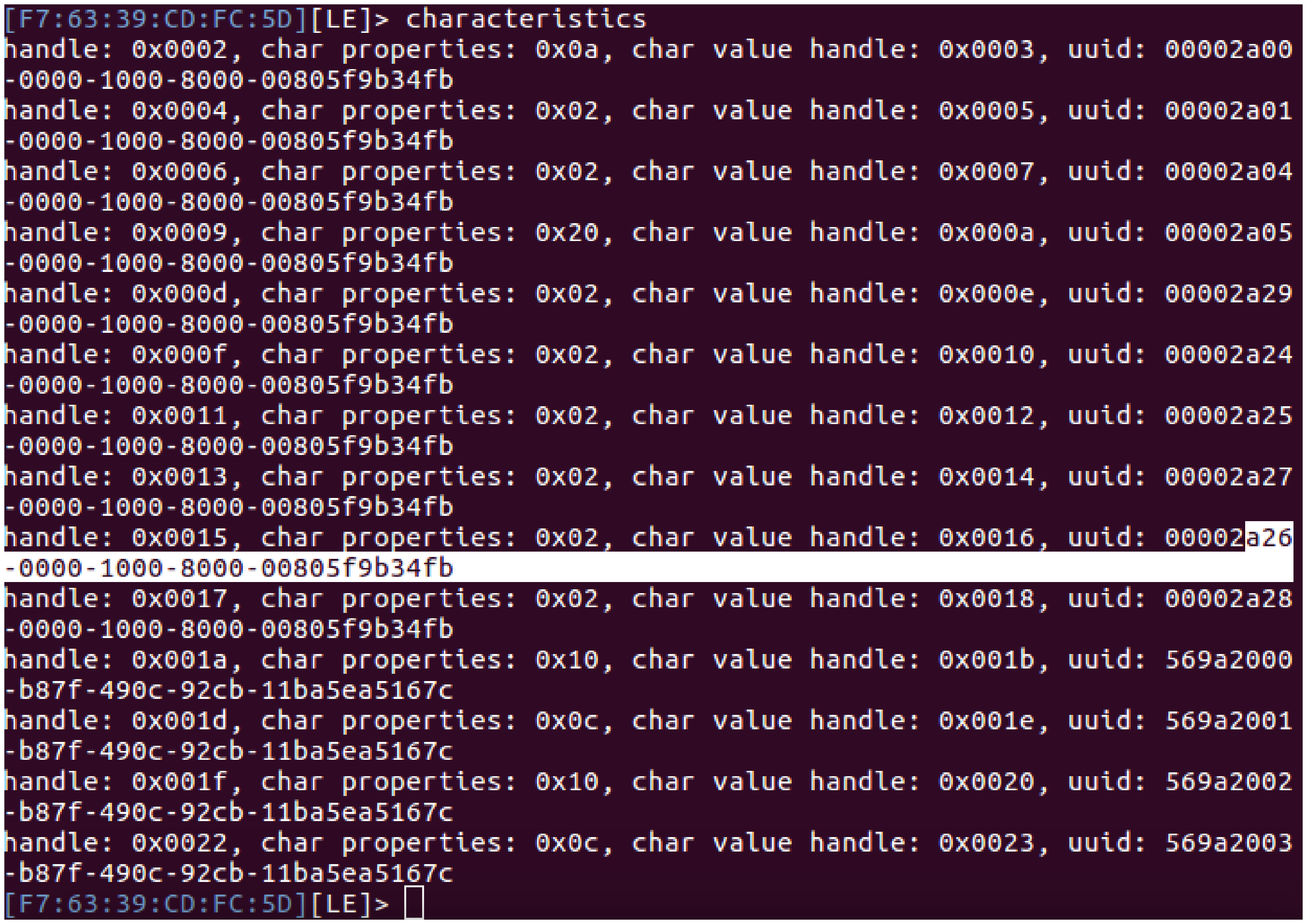
The Linux computer returns *Connection successful* and the BT860 connected to UwTerminal reports the connection as well (Figure 12).



```
test@test-ThinkPad-T60p:~$ gatttool -b F7:63:39:CD:FC:5D -t random -I
[F7:63:39:CD:FC:5D][LE]> Connect
Attempting to connect to F7:63:39:CD:FC:5D
Connection successful
```

Figure 12: Make a BLE connection to the module

To locate the handles for TX and RX, send the command *characteristics* to obtain the list of characteristics (including the properties, handle value, and UUID). From the BL600 *smart*BASIC extension guide, the BL600 TX characteristic UUID is *569a2000-b87f-490c-92cb-11ba5ea5167c* and the Linux host must enable notification to receive data from the BL600. The BL600 RX characteristic UUID is *569a2001-b87f-490c-92cb-11ba5ea5167c* and the Linux host writes to it (Figure 13).

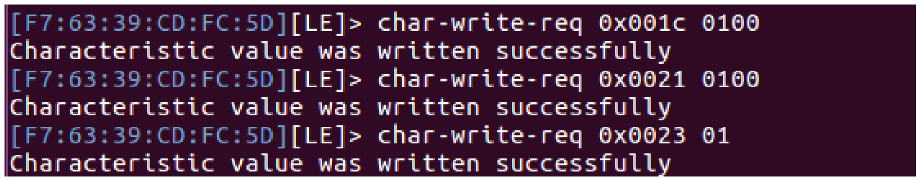


```
[F7:63:39:CD:FC:5D][LE]> characteristics
handle: 0x0002, char properties: 0x0a, char value handle: 0x0003, uuid: 00002a00
-0000-1000-8000-00805f9b34fb
handle: 0x0004, char properties: 0x02, char value handle: 0x0005, uuid: 00002a01
-0000-1000-8000-00805f9b34fb
handle: 0x0006, char properties: 0x02, char value handle: 0x0007, uuid: 00002a04
-0000-1000-8000-00805f9b34fb
handle: 0x0009, char properties: 0x20, char value handle: 0x000a, uuid: 00002a05
-0000-1000-8000-00805f9b34fb
handle: 0x000d, char properties: 0x02, char value handle: 0x000e, uuid: 00002a29
-0000-1000-8000-00805f9b34fb
handle: 0x000f, char properties: 0x02, char value handle: 0x0010, uuid: 00002a24
-0000-1000-8000-00805f9b34fb
handle: 0x0011, char properties: 0x02, char value handle: 0x0012, uuid: 00002a25
-0000-1000-8000-00805f9b34fb
handle: 0x0013, char properties: 0x02, char value handle: 0x0014, uuid: 00002a27
-0000-1000-8000-00805f9b34fb
handle: 0x0015, char properties: 0x02, char value handle: 0x0016, uuid: 00002a26
-0000-1000-8000-00805f9b34fb
handle: 0x0017, char properties: 0x02, char value handle: 0x0018, uuid: 00002a28
-0000-1000-8000-00805f9b34fb
handle: 0x001a, char properties: 0x10, char value handle: 0x001b, uuid: 569a2000
-b87f-490c-92cb-11ba5ea5167c
handle: 0x001d, char properties: 0x0c, char value handle: 0x001e, uuid: 569a2001
-b87f-490c-92cb-11ba5ea5167c
handle: 0x001f, char properties: 0x10, char value handle: 0x0020, uuid: 569a2002
-b87f-490c-92cb-11ba5ea5167c
handle: 0x0022, char properties: 0x0c, char value handle: 0x0023, uuid: 569a2003
-b87f-490c-92cb-11ba5ea5167c
[F7:63:39:CD:FC:5D][LE]>
```

Figure 13: List of characteristics

To enable notification, send the following command:

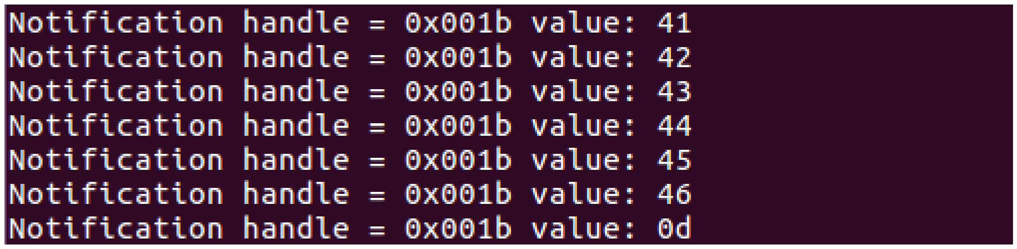
```
char-write-req 0x001c 010
```



```
[F7:63:39:CD:FC:5D][LE]> char-write-req 0x001c 0100
Characteristic value was written successfully
[F7:63:39:CD:FC:5D][LE]> char-write-req 0x0021 0100
Characteristic value was written successfully
[F7:63:39:CD:FC:5D][LE]> char-write-req 0x0023 01
Characteristic value was written successfully
```

Figure 14: Enable notification for Modem-In and TX characteristics and written to Modem-Out characteristic to set the value to 1

From UwTerminal, you can enter *ABCDEF* into the terminal and press **Enter**. Data is received on the Linux computer.

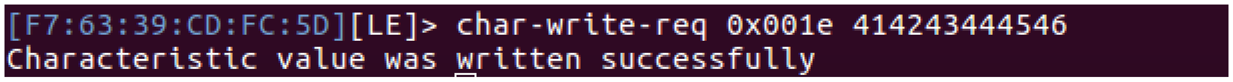


```
Notification handle = 0x001b value: 41
Notification handle = 0x001b value: 42
Notification handle = 0x001b value: 43
Notification handle = 0x001b value: 44
Notification handle = 0x001b value: 45
Notification handle = 0x001b value: 46
Notification handle = 0x001b value: 0d
```

Figure 15: ABCDEF\n sent and received as notification

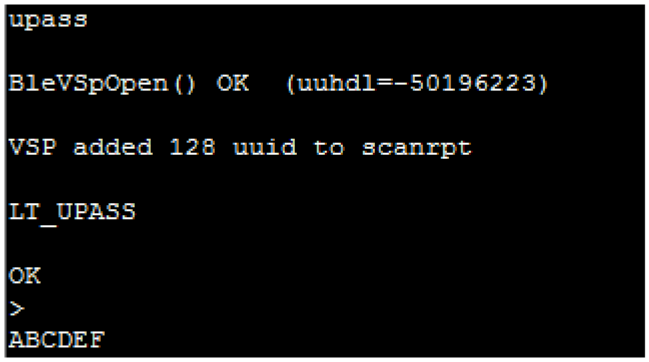
To send from the BT860 side with *ABCDEF* as data, enter the following command:

```
char-write-cmd  0x001e 414243444546
```



```
[F7:63:39:CD:FC:5D][LE]> char-write-req 0x001e 414243444546
Characteristic value was written successfully
```

Figure 16: Writing ABCDEF to the module



```
upass

BleVSpOpen() OK  (uuhdl=-50196223)

VSP added 128 uuid to scanrpt

LT_UPASS

OK
>
ABCDEF
```

Figure 17: Data ABCDEF is received on the module side

Revision History

Version	Date	Notes	Contributors	Approver
1.0	18 Jan 2018	Initial Release	Raymond Au	Jonathan Kaye