

# HCI Bluetooth Module SPP Connection on Linux

Application Note

v1.0

## INTRODUCTION

Laird offers a wide range of HCI Bluetooth modules supporting UART and USB interfaces with a chipset from different vendors. Most people use these modules on Linux running the BlueZ stack. This application note describes how to make the SPP connection with these HCI modules. The SPP connection should work for BT800, BT820, BT830, BT850, BT851, BT860, and uB2.

## REQUIREMENTS

- BT851 USB dongle
- BlueZ – Official Linux Bluetooth protocol stack
- BTM411 Laird Bluetooth module
- Laird Ezurio Terminal for Windows PC
- Picocom

**Notes:** BT851 is a USB dongle. The USB driver should be available in the kernel. Ubuntu 16.04 is used as the testing platform (Kernel version 4.4.0-31). The BlueZ stack (v 5.37) is included.

BTM411 is a built-in stack BT module from Laird. It is operating by AT commands which make it easier for demonstration.

Laird Ezurio Terminal is a serial terminal. It can be downloaded at this link  
[https://assets.lairdtech.com/home/brandworld/files/LairdTerminal\\_v6\\_9\\_0.zip](https://assets.lairdtech.com/home/brandworld/files/LairdTerminal_v6_9_0.zip).

Picocom is serial Terminal for Linux.

## PREPARATION

After inserting the BT851 USB dongle, confirm that the radio status is ready by typing `hciconfig` (Figure 1). The host should be able to detect and recognize the BT dongle once it is inserted. If you are using a UART interface module, you must attach the radio before the host can recognize it. Please refer to the module page for details. Superuser permissions are required for the Bluetooth-related operation on the Linux.

```
test@test-ThinkPad-T60p:~$ hciconfig
hci0: Type: BR/EDR Bus: USB
      BD Address: 00:17:23:00:00:01 ACL MTU: 1021:8 SCO MTU: 64:1
      UP RUNNING
      RX bytes:861 acl:0 sco:0 events:71 errors:0
      TX bytes:5067 acl:0 sco:0 commands:71 errors:0
```

Figure 1: Check BT851 status with “hciconfig”

The Bluetooth module is first configured as discoverable and connectable and configured to auto-answer an incoming SPP connection (Figure 2).

```
Open [COM73:9600,N,8,1]
at
OK
ats512=4
OK
ats0=1
OK
at&w
OK
atz
<<Modem Status : All DEASSERTED >>
OK
ati
Laird Bluetooth Data Module 411
OK
```

Figure 2: Configure the BT module to be discoverable and connectable and to auto-answer on one ring

Use the `sudo apt search picocom` command to determine if picocom is installed (Figure 3). If it's not installed, enter `sudo apt install picocom` to install it.

```
test@test-ThinkPad-T60p:~$ sudo apt search picocom
Sorting... Done
Full Text Search... Done
picocom/xenial-updates,xenial-security,now 1.7-2build0.16.04.1 i386 [installed]
minimal dumb-terminal emulation program
```

Figure 3: Shows that picocom is already installed on this Linux machine

## CLASSIC BT SCAN

The following is the command to initialize a scan.

```
hcitool scan
```

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

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

```
test@test-ThinkPad-T60p:~$ hcitool scan
Scanning ...
00:16:A4:0B:06:A0   Laird BTM 0B06A0
7C:7D:3D:50:17:DC   HUawei WATCH 0570
```

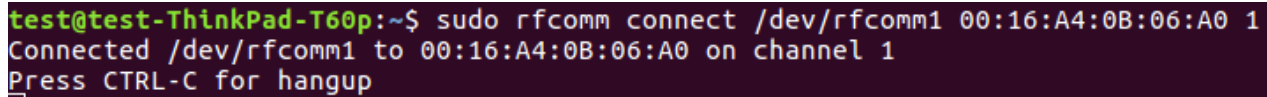
Figure 4: Scan for nearby BT devices

## OUTGOING SPP CONNECTION

To make a connection, issue the following command:

```
sudo rfcomm connect /dev/rfcomm1 <MAC_ADDRESS> 1
```

`/dev/rfcomm1` is a symbolic link which is auto-created if the connection is successful (Figure 5).

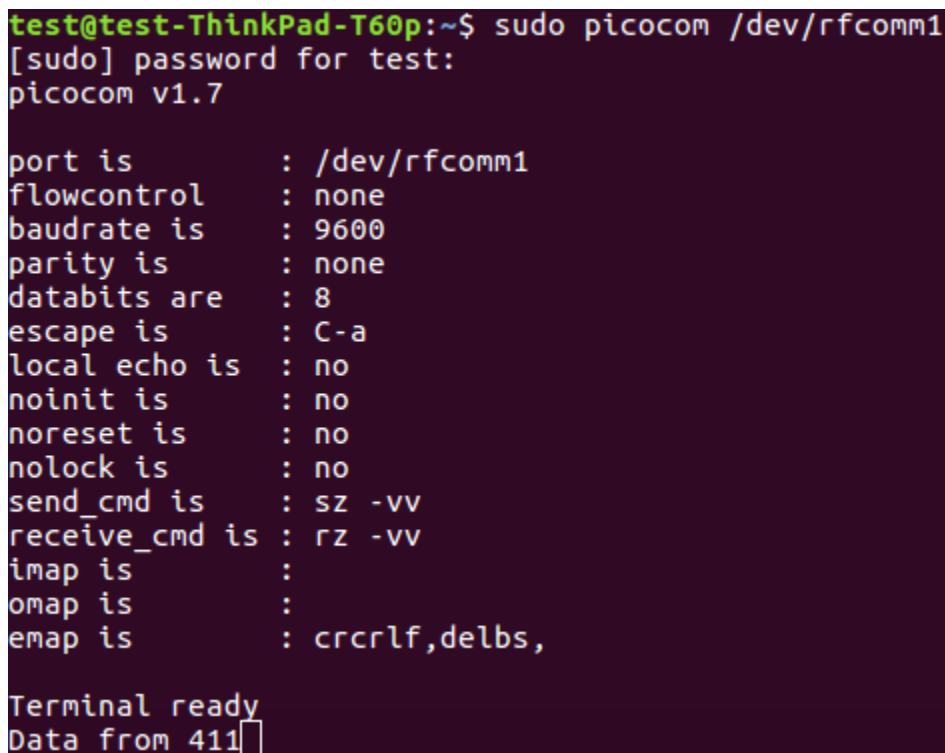


```
test@test-ThinkPad-T60p:~$ sudo rfcomm connect /dev/rfcomm1 00:16:A4:0B:06:A0 1
Connected /dev/rfcomm1 to 00:16:A4:0B:06:A0 on channel 1
Press CTRL-C for hangup
```

Figure 5: Make a Rfcomm connection to the module

After the connection is established, open a symbolic link by using the following command:

```
picocom symbolic link
```



```
test@test-ThinkPad-T60p:~$ sudo picocom /dev/rfcomm1
[sudo] password for test:
picocom v1.7

port is          : /dev/rfcomm1
flowcontrol     : none
baudrate is     : 9600
parity is       : none
databits are    : 8
escape is       : C-a
local echo is   : no
noinit is      : no
noreset is     : no
nolock is       : no
send_cmd is    : SZ -vv
receive_cmd is : rZ -vv
imap is        :
omap is        :
emap is        : crCrLf,delbs,

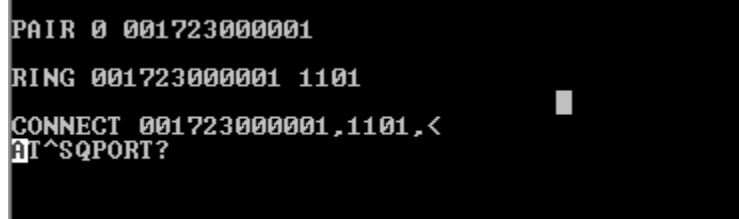
Terminal ready
Data from 411
```

Figure 6: Open `/dev/rfcomm1` with `picocom`

## SUPPRESSING THE MODEM QUERY COMMANDS

Once the RFcomm connection is made, the Linux modem manager may send the following unsolicited commands to check if a modem is connected over the Bluetooth connection.

```
AT^SQPORT?  
AT  
AT  
AT
```



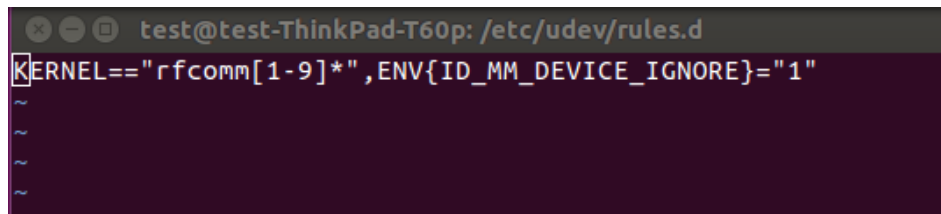
```
PAIR 0 001723000001  
RING 001723000001 1101  
CONNECT 001723000001,1101,<  
AT^SQPORT?
```

Figure 7: Unsolicited modem query commands received by the module

To prevent the modem manager from sending these modem query commands, you can set a rule with the *udev* device manager in the `/etc/udev/rules.d` directory by creating the following file:

```
90-rfcomm.rules
```

Add it below the line indicated below.



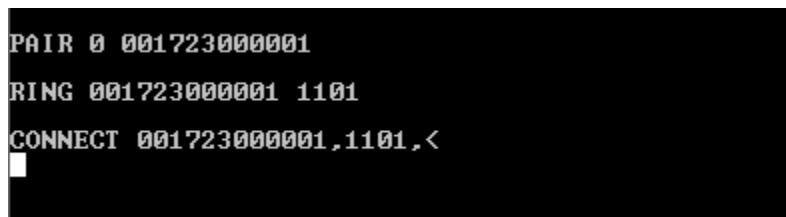
```
test@test-ThinkPad-T60p: /etc/udev/rules.d  
KERNEL=="rfcomm[1-9]*",ENV{ID_MM_DEVICE_IGNORE}="1"  
~  
~  
~
```

Figure 8: Added rule to ignore the modem device on rfcomm 1 to 9

You can either reboot your device or reload the rules.

Use the device manager to reload the rule:

```
sudo udevadm control -reload-rules
```



```
PAIR 0 001723000001  
RING 001723000001 1101  
CONNECT 001723000001,1101,<  
█
```

Figure 9: Suppressed modem query commands

## INCOMING SPP CONNECTION

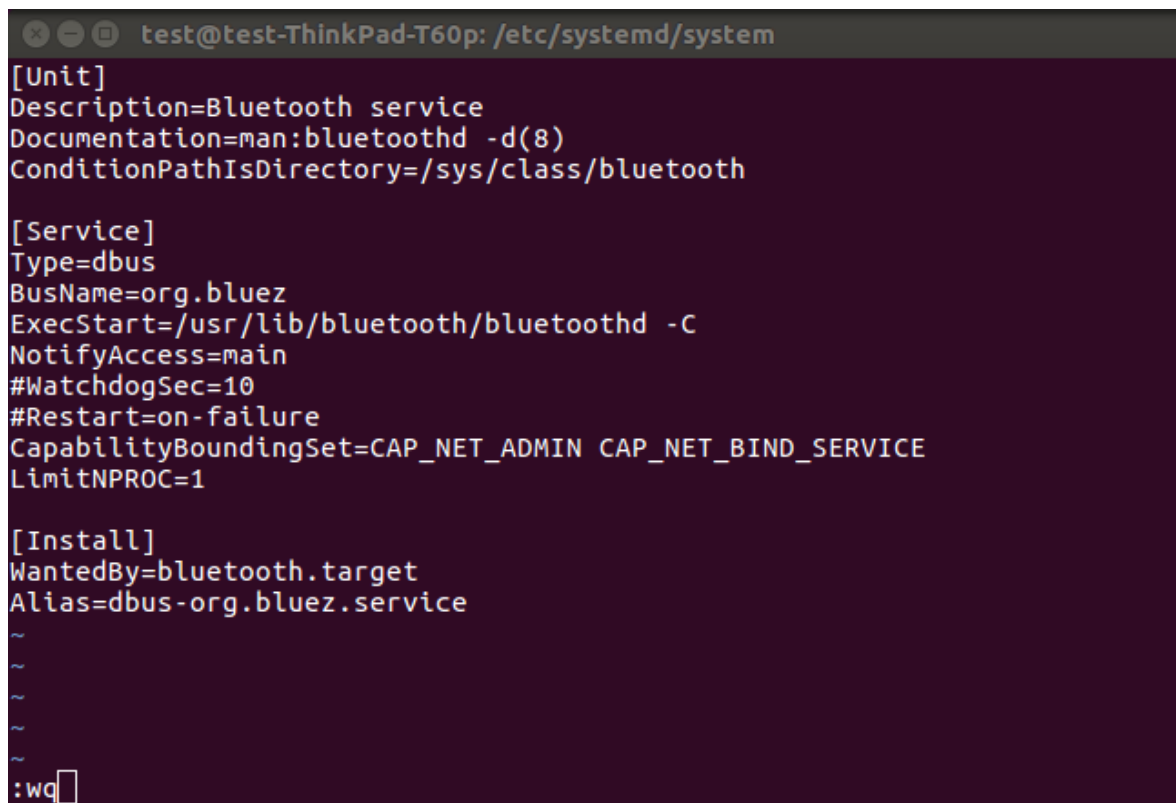
By default, the Linux Bluetooth is not configured as discoverable and connectable. The `hciconfig hci0 piscan` command should turn on these modes and the module will be able to obtain the Linux BT MAC address via scan. It is important to turn discoverable mode off when it's not needed by issuing the `hciconfig hci0 noscan` command. After pairing, issue the `hciconfig hci0 piscan` command to remain connectable but not discoverable.

The SP (Serial Port) service is added by the sdptool. If you are running on BlueZ 5.X, you may encounter the following error. If so, you cannot browse the local existing service.

```
test@test-ThinkPad-T60p:~$ sudo sdptool browse local
Failed to connect to SDP server on FF:FF:FF:00:00:00: No such file or directory
```

Figure 10: Failure to browse local service

To solve this issue, enable the *command line interface* in the `/etc/systemd/system/dbus-org.bluez.service` file when the Bluetooth service starts.



```
test@test-ThinkPad-T60p: /etc/systemd/system
[Unit]
Description=Bluetooth service
Documentation=man:bluetoothd -d(8)
ConditionPathIsDirectory=/sys/class/bluetooth

[Service]
Type=dbus
BusName=org.bluez
ExecStart=/usr/lib/bluetooth/bluetoothd -C
NotifyAccess=main
#WatchdogSec=10
#Restart=on-failure
CapabilityBoundingSet=CAP_NET_ADMIN CAP_NET_BIND_SERVICE
LimitNPROC=1

[Install]
WantedBy=bluetooth.target
Alias=dbus-org.bluez.service
~
~
~
~
~
:wq
```

Figure 11: Enable command line interface with -C

Reload the daemon and restart Bluetooth. You can now browse the local Bluetooth Service and add the SP (Serial Port) service. The auto-assigned channel number is 1.

```
test@test-ThinkPad-T60p:/etc/systemd/system$ sudo systemctl daemon-reload
test@test-ThinkPad-T60p:/etc/systemd/system$ sudo systemctl restart bluetooth
```

Figure 12: Reload daemon and restart Bluetooth

```
test@test-ThinkPad-T60p:/etc/systemd/system$ sudo sdptool browse local
Browsing FF:FF:FF:00:00:00 ...
Service RecHandle: 0x10000
Service Class ID List:
  "PnP Information" (0x1200)
Profile Descriptor List:
  "PnP Information" (0x1200)
  Version: 0x0103

Browsing FF:FF:FF:00:00:00 ...
Service Search failed: Invalid argument
Service Name: Generic Access Profile
Service Provider: BlueZ
```

Figure 13: sdptool can browse local service

```
test@test-ThinkPad-T60p:/etc/systemd/system$ sudo sdptool add SP
Serial Port service registered
```

Figure 14: Add Serial Port service

```
Service Name: Serial Port
Service Description: COM Port
Service Provider: BlueZ
Service RecHandle: 0x10010
Service Class ID List:
  "Serial Port" (0x1101)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
  Channel: 1
Language Base Attr List:
  code_ISO639: 0x656e
  encoding: 0x6a
  base_offset: 0x100
Profile Descriptor List:
  "Serial Port" (0x1101)
  Version: 0x0100
```

Figure 15: Serial port is added at channel 1

You can now listen for an incoming SPP connection.

```
test@test-ThinkPad-T60p:~$ sudo rfcomm listen /dev/rfcomm1 1
[sudo] password for test:
Waiting for connection on channel 1
Connection from 00:16:A4:0B:06:A0 to /dev/rfcomm1
Press CTRL-C for hangup
```

Figure 16: Listen on channel 1

## REVISION HISTORY

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