



Bluetooth™ Protocol Stack

For STM3240G

SPPLE-DEMO Sample Application

User Manual

Version: 2.1.3
January 4th, 2010



Louisville, KY

www.stonestreetone.com

Bluetooth and the Bluetooth logos are trademarks owned by Bluetooth SIG, Inc., USA and licensed to Stonestreet One. Bluetopia®, Stonestreet One™, and the Stonestreet One logo are registered trademarks of Stonestreet One, Louisville, Kentucky, USA. All other trademarks are property of their respective owners.

Copyright © 2000-2009 by Stonestreet One. All rights reserved.

Table of Contents

DEMO OVERVIEW	3
Running the Bluetooth Code.....	3
DEMO APPLICATION	4
Device 1 (server) setup on the demo application.....	4
Device 2 (client) setup on the demo application	5
Initiating connection from device 2	6
Identify supported services.....	7
Data Transfer between Client and Server	9
Connecting to an iPhone running SPPLE Chat application.....	11

Demo Overview

This application demonstrates a BR/EDR SPP based application as well as a custom application, SPPLLE, over Bluetooth LE that is similar in functionality to the BR/EDR application. The SPPLLE Profile is similar to the SPP profile except that it uses LE transport compared to BR/EDR transport in the SPP profile.

The SPP profile emulates serial cable connections. There are two roles defined in this profile. The first is the server that has the SPPLLE service running on it and has open a server port. The client is a device that connects to the server. Both of these devices can then exchange data with each other.

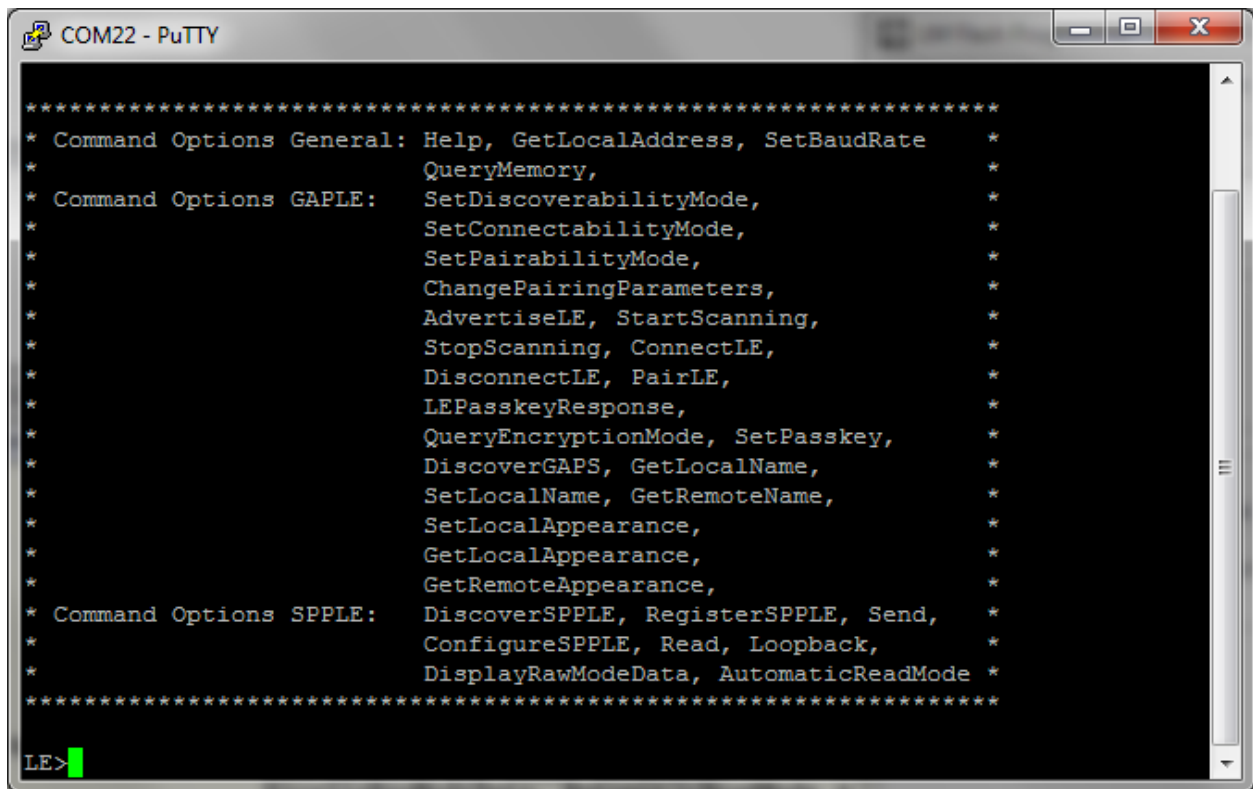
This document talks about the SPPLLE application in details.

The application allows the user to use a console to use Bluetooth Low Energy (BLE) to establish connection between two BLE devices, send Bluetooth commands and exchange data over BLE.

The first part of this document explains how to connect to two STM3240G boards over SPPLLE and transfer data. The second part talks about how to connect the STM3240G to an iPhone application.

Running the Bluetooth Code

Once the code is flashed, connect the board to a PC using a serial cable (or serial or to USB converter). Once connected, wait for the driver to install. Attach a Terminal program like Tera Term to the serial port x (the serial port you connected your hardware to) for the board. The serial parameter to use is 115200 Baud, 8, no, 1, no flow control. Once connected, reset the device using Reset button (located next to the display and you should see the stack getting initialized on the terminal and the help screen will be displayed, which shows all of the commands.



```
*****
* Command Options General: Help, GetLocalAddress, SetBaudRate *
* QueryMemory, *
* Command Options GAPLE: SetDiscoverabilityMode, *
* SetConnectabilityMode, *
* SetPairabilityMode, *
* ChangePairingParameters, *
* AdvertiseLE, StartScanning, *
* StopScanning, ConnectLE, *
* DisconnectLE, PairLE, *
* LEPasskeyResponse, *
* QueryEncryptionMode, SetPasskey, *
* DiscoverGAPS, GetLocalName, *
* SetLocalName, GetRemoteName, *
* SetLocalAppearance, *
* GetLocalAppearance, *
* GetRemoteAppearance, *
* Command Options SPPLE: DiscoverSPPLE, RegisterSPPLE, Send, *
* ConfigureSPPLE, Read, Loopback, *
* DisplayRawModeData, AutomaticReadMode *
*****
LE>
```

Now connect the second board via another serial cable and follow the same steps performed before when running the Bluetooth code on the first board.

Demo Application

The demo application provides a description on how to use the demo application to connect two configured boards and communicate over bluetoothLE. The included application registers a custom service on a board when the stack is initialized.

Device 1 (server) setup on the demo application

a) To start with one of the devices has to have the SPP-LE Service running on it. It can be started by running RegisterSPPLE.

```
* SetPairabilityMode, *
* ChangePairingParameters, *
* AdvertiseLE, StartScanning, *
* StopScanning, ConnectLE, *
* DisconnectLE, PairLE, *
* LEPasskeyResponse, *
* QueryEncryptionMode, SetPasskey, *
* DiscoverGAPS, GetLocalName, *
* SetLocalName, GetRemoteName, *
* SetLocalAppearance, *
* GetLocalAppearance, *
* GetRemoteAppearance, *
* Command Options SPPLE: DiscoverSPPLE, RegisterSPPLE, Send, *
* ConfigureSPPLE, Read, Loopback, *
* DisplayRawModeData, AutomaticReadMode *
*****
LE>RegisterSPPLE
Successfully registered SPPLE Service. a)
LE>AdvertiseLE 1
GAP_LE_Advertising_Enable success. b)
LE>
```

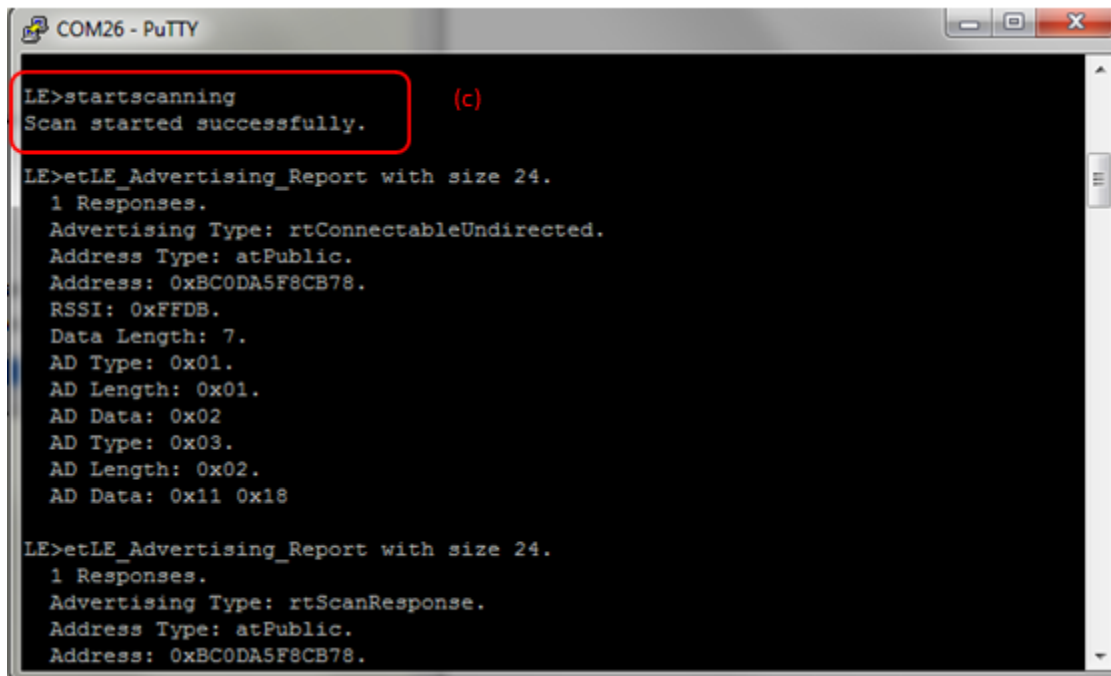
b) Next, the device acting as a server needs to advertise to other devices. This can be done by running AdvertiseLE 1.

Device 2 (client) setup on the demo application

[Steps c and d are optional if you already know the Bluetooth address of the device that you want to connect to]

c) The client LE device can try to find which LE devices are in the vicinity using the command: StartScanning.

d) Once you have found the device, you can stop scanning by using the command: StopScanning.



```
COM26 - PuTTY
LE>startscanning
Scan started successfully.
(c)
LE>etLE_Advertising_Report with size 24.
1 Responses.
Advertising Type: rtConnectableUndirected.
Address Type: atPublic.
Address: 0xBC0DA5F8CB78.
RSSI: 0xFFDB.
Data Length: 7.
AD Type: 0x01.
AD Length: 0x01.
AD Data: 0x02
AD Type: 0x03.
AD Length: 0x02.
AD Data: 0x11 0x18
LE>etLE_Advertising_Report with size 24.
1 Responses.
Advertising Type: rtScanResponse.
Address Type: atPublic.
Address: 0xBC0DA5F8CB78.
```

Initiating connection from device 2

e) Once the application on the client side knows the Bluetooth address of the device that is advertising, it can connect to that device using the command: `ConnectLE <Bluetooth Address>`

```
COM26 - PuTTY
*
*      AdvertiseLE, StartScanning,
*      StopScanning, ConnectLE,
*      DisconnectLE, PairLE,
*      LEPasskeyResponse,
*      QueryEncryptionMode, SetPasskey,
*      DiscoverGAPS, GetLocalName,
*      SetLocalName, GetLERemoteName,
*      SetLocalAppearance,
*      GetLocalAppearance,
*      GetRemoteAppearance,
* Command Options SPPLE: DiscoverSPPLE, RegisterSPPLE, LESend,
*      ConfigureSPPLE, LERead, Loopback,
*      DisplayRawModeData, AutomaticReadMode
*
*****
SPP+LE>ConnectLE BC0DA5F8BDC6      (e)
Connection Request successful.

SPP+LE>etLE_Connection_Complete with size 18.
Status:      0x00.
Role:        Master.
Address Type: Public.
BD_ADDR:     0xBC0DA5F8BDC6.

SPP+LE>
etGATT_Connection_Device_Connection with size 12:
Connection ID: 1.
Connection Type: LE.
Remote Device: 0xBC0DA5F8BDC6.
Connection MTU: 23.

SPP+LE>
Exchange MTU Response.
Connection ID: 1.
Transaction ID: 1.
Connection Type: LE.
BD_ADDR:      0xBC0DA5F8BDC6.
MTU:          48.

SPP+LE>
SPP+LE>
```

Identify supported services

f) After Initialization, the device needs to find out if SPP services are supported. To do this DiscoverSPPLE is run on the client.

```
COM26 - PuTTY
* Command Options SPPL: DiscoverSPPLE, RegisterSPPLE, LERSend, *
* ConfigureSPPLE, LERead, Loopback, *
* DisplayRawModeData, AutomaticReadMode *
*****

SPP+LE>ConnectLE BC0DA5F8BDC6
Connection Request successful.

SPP+LE>etLE_Connection_Complete with size 18.
  Status: 0x00.
  Role: Master.
  Address Type: Public.
  BD_ADDR: 0xBC0DA5F8BDC6.

SPP+LE>
etGATT_Connection_Device_Connection with size 12:
  Connection ID: 1.
  Connection Type: LE.
  Remote Device: 0xBC0DA5F8BDC6.
  Connection MTU: 23.

SPP+LE>
Exchange MTU Response.
  Connection ID: 1.
  Transaction ID: 1.
  Connection Type: LE.
  BD_ADDR: 0xBC0DA5F8BDC6.
  MTU: 48.

SPP+LE>
SPP+LE>DiscoverSPPLE
GDIS_Service_Discovery_Start success.

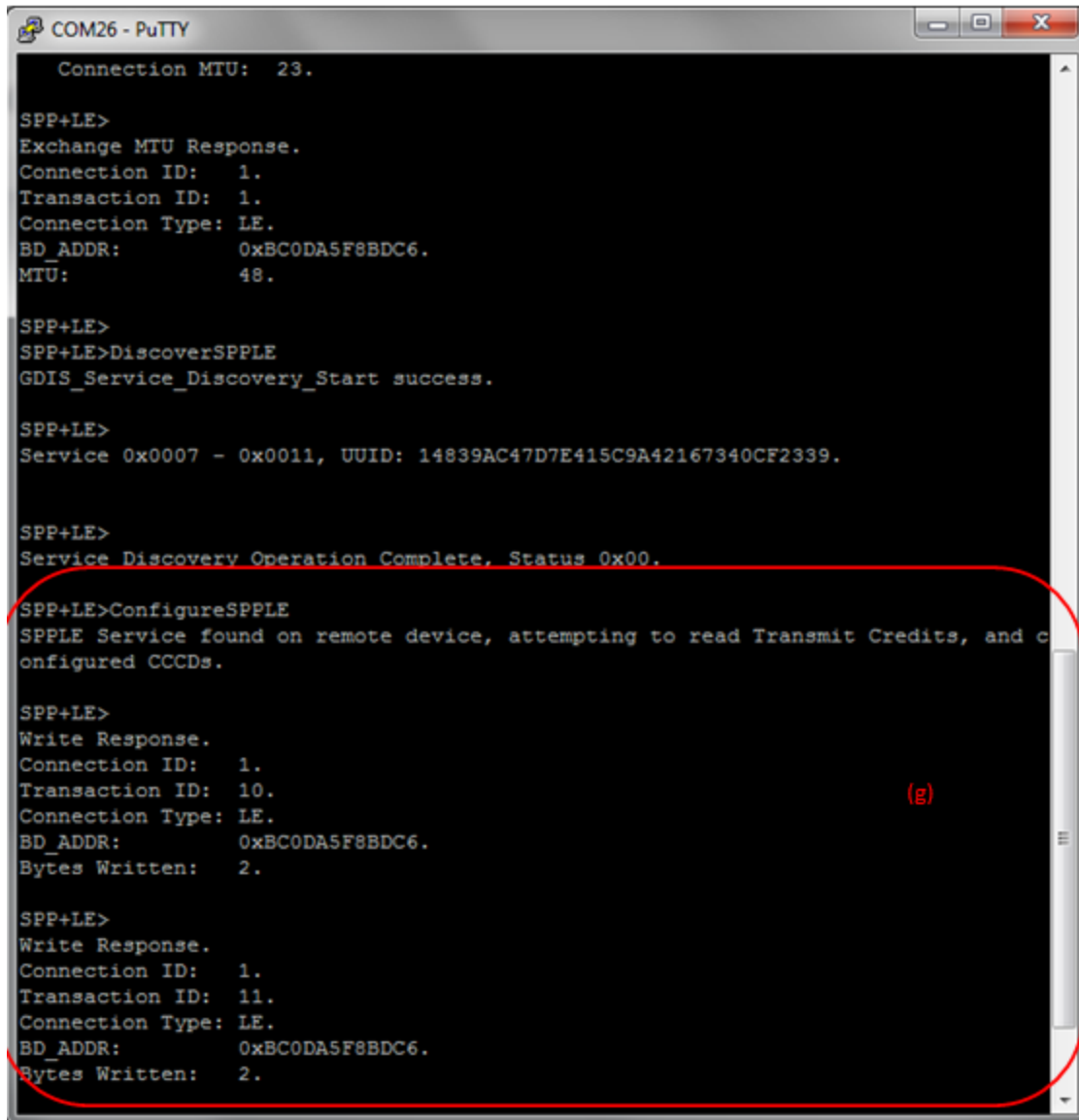
SPP+LE>
Service 0x0007 - 0x0011, UUID: 14839AC47D7E415C9A42167340CF2339.

SPP+LE>
Service Discovery Operation Complete, Status 0x00.

SPP+LE>
```

(f)

g) After finding out support for SPP-LE, we need to configure SPP-LE. This is done by running ConfigureSPPLE on the client.



```
COM26 - PuTTY
Connection MTU: 23.

SPP+LE>
Exchange MTU Response.
Connection ID: 1.
Transaction ID: 1.
Connection Type: LE.
BD_ADDR: 0xBC0DA5F8BDC6.
MTU: 48.

SPP+LE>
SPP+LE>DiscoverSPPLE
GDIS_Service_Discovery_Start success.

SPP+LE>
Service 0x0007 - 0x0011, UUID: 14839AC47D7E415C9A42167340CF2339.

SPP+LE>
Service Discovery Operation Complete. Status 0x00.

SPP+LE>ConfigureSPPLE
SPPLE Service found on remote device, attempting to read Transmit Credits, and c
onfigured CCCDs.

SPP+LE>
Write Response.
Connection ID: 1.
Transaction ID: 10.
Connection Type: LE.
BD_ADDR: 0xBC0DA5F8BDC6.
Bytes Written: 2.

SPP+LE>
Write Response.
Connection ID: 1.
Transaction ID: 11.
Connection Type: LE.
BD_ADDR: 0xBC0DA5F8BDC6.
Bytes Written: 2.
```

Data Transfer between Client and Server

h) After configuring we can send data between client and server. To send data we use Senddata <number of bytes>.

COM22:115200baud - Tera Term VT

File Edit Setup Control Window Help

```
LE>configure
Service 0x000A - 0x0010, UUID: 14839AC47D7E415C9A42167340CF2339.

LE>
Service Discovery Operation Complete, Status 0x00.

LE>sppl
SPPLE Service found on remote device, attempting to read Transmit Credits, and c

LE>
Write Response.
Connection ID: 1.
Transaction ID: 11.
Connection Type: LE.
BD_ADDR: 0x444E68A4FDD8.
Bytes Written: 2.

LE>senddata 100
Send Complete, Sent 100.

LE>
LE>
```

h)

- i) Once the other device receives the data it receives a Data Indication event.
- j) The receiving device can then read the data that was sent using command: Read

COM22:115200baud - Tera Term VT

File Edit Setup Control Window Help

```
LE>
LE>
Data Indication Event, Connection ID 1, Received 11 bytes.

LE>
Data Indication Event, Connection ID 1, Received 3 bytes.

LE>read
Read: 14.
: test textEOM

LE>
```

j)

k) This will print out the data that was sent. This data was sent over BluetoothLE using a custom service of SPPL in the sample application.

l) AutomaticReadMode makes sure that all data received by the device is read automatically (instead of manually using LERead as described above).

m) DisplayRawModeData displays the data that has been read automatically.

n) Loopback is used to send back the data that is being sent by the remote device. So if the remote device wrote 50 characters, turning on Loopback would send those characters back to the remote device where it can be read.

Connecting to an iPhone running SPPL Chat application

The STM device can also connect to an iPhone running an SPPL application. The application that we use on the iPhone is SPPL Chat which can be downloaded for free from the app store [here](#). There are some changes that need to be made to the SPPLDemo.C file.

- 1) In the function ConnectLEDevice make the following changes (to the result = statement) after the */* Everything appears correct, go ahead and attempt to make the connection. */* comment and before the *if(!Result)* statement.

```
Result = GAP_LE_Create_Connection(BluetoothStackID, 100, 100,
Result?fpNoFilter:fpWhiteList, latRandom, Result?&BD_ADDR:NULL, latPublic,
&ConnectionParameters, GAP_LE_Event_Callback, 0);
```

- 2) In the function ConfigureSPPL make the following changes (to the if statement) after the */* Determine if a service discovery operation has been previously done */* comment and before the else case.

```
if(TRUE)
{
    Display("SPPL Service found on remote device, attempting to
read Transmit Credits, and configured CCCDs.\r\n");

    /* Enable Notifications on the proper characteristics. */

    EnableDisableNotificationsIndications(DeviceInfo->ClientInfo.Tx_Client_Configuration_Descriptor,
GATT_CLIENT_CONFIGURATION_CHARACTERISTIC_NOTIFY_ENABLE,
GATT_ClientEventCallback_SPPL);

    ret_val = 0;
}
```

- 3) In the function SendDataCommand add the following code after the SendInfo.BytesSent = 0 and before the /* Kick start the send process. */ comment, add the following code.

```
DeviceInfo->TransmitCredits = 100;  
DeviceInfo->ServerInfo.Tx_Client_Configuration_Descriptor =  
GATT_CLIENT_CONFIGURATION_CHARACTERISTIC_NOTIFY_ENABLE;
```

- 4) Load the SPP LE profile on to the STM3240G device by rebuilding the project and flashing it from the project.
- 5) Set up a Terminal Program for the Serial Port that the device is connected to. The serial parameters to use are 115200 Baud, 8, no, 1 and no flow control. Once connected, reset the device using Reset S3 button and you should see the stack getting initialized on the terminal.
- 6) On the iPhone open the SPPLE chat application. Choose peripheral mode and turn on advertising.
- 7) On the STM device, StartScanning to find out devices in the area that are connectable. The Bluetooth address of the Iphone should show up something like this:

```
etLE_Advertising_Report with size 36.  
1 Responses.  
Advertising Type: rtConnectableUndirected.  
Address Type: atRandom.  
Address: 0x79F20C012372.  
RSSI: 0xFFFFF7CB.  
Data Length: 29.  
AD Type: 0x01.  
AD Length: 0x01.  
AD Data: 0x1A  
AD Type: 0x07.  
AD Length: 0x10.  
AD Data: 0x39 0x23 0xCF 0x40 0x73 0x16 0x42 0x9A 0x5C 0x41 0x7E 0x7D  
0xC4 0x9A 0x83 0x14  
AD Type: 0x09.  
AD Length: 0x06.  
AD Data: 0x69 0x50 0x68 0x6F 0x6E 0x65
```

```
LE>etLE_Advertising_Report with size 36.  
1 Responses.  
Advertising Type: rtScanResponse.  
Address Type: atRandom.  
Address: 0x79F20C012372.  
RSSI: 0xFFFFF7CB.  
Data Length: 0.
```

- 8) The address type will be random. Note down the address of the device specified.
- 9) Connect to the remote device using the connectle <bd-addr> command where the bd-addr is the previously noted address.

- 10) Discover services using discoverspple and configure services using configurespple.
- 11) Now the two devices are connected. Data from the iphone can be send by typing text on the text box and hitting send.
- 12) Data from the STM device can be sent using the senddata command. It is read and displayed automatically in the output window of the app.

