# BL652 Data Length Extension

## BL652

*Application Note* *v1.1*

## INTRODUCTION

The goals of this document include the following:

- Demonstrate how to run speed tests between two BL652 devices using UwTerminalX.
- Give an overview of how Data Packet Length Extension feature is used to improve the throughput.
- Demonstrate how the Data Packet Length Extension and high bandwidth features can be enabled on the BL652.

## REQUIREMENTS

- 2 x Laird DVK-BL652 (or equivalent RS-232 to serial connections to two BL652 Modules)
- FTDI USB to Serial drivers for DVK-BL652 (found at http://www.ftdichip.com/FTDrivers.htm)
- 2 x USB-A to USB-Micro cable
- UwTerminalX (version 1.08n or later), available at https://github.com/LairdCP/UwTerminalX/releases
- BL652 firmware version 28.6.2.0, found in the BL652 Software Downloads tab at https://www.lairdtech.com/products/bl652-ble-module
- The following *smart*BASIC applications found in https://github.com/LairdCP/BL652-Applications:
  - $autorun$.dle.data.length.extension.peripheral.sb
  - $autorun$.dle.data.length.extension.central.sb

## BIDIRECTIONAL TEST SEQUENCE

**IMPORTANT!**
Currently, Android and iOS have limited support for the Bluetooth v4.2 LE Data Length Extension and High Bandwidth Configuration features. To see the full benefits of these two features' impact on throughput, two BL652 kits are needed for testing.

To test the BLE link's throughput between two BL652 devices, perform the following steps:-

1. Connect the two BL652 devices to the PC. Please ensure that you use the correct USB port and that J12 is connected to pins 1&2 as shown in Figure 1:



Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

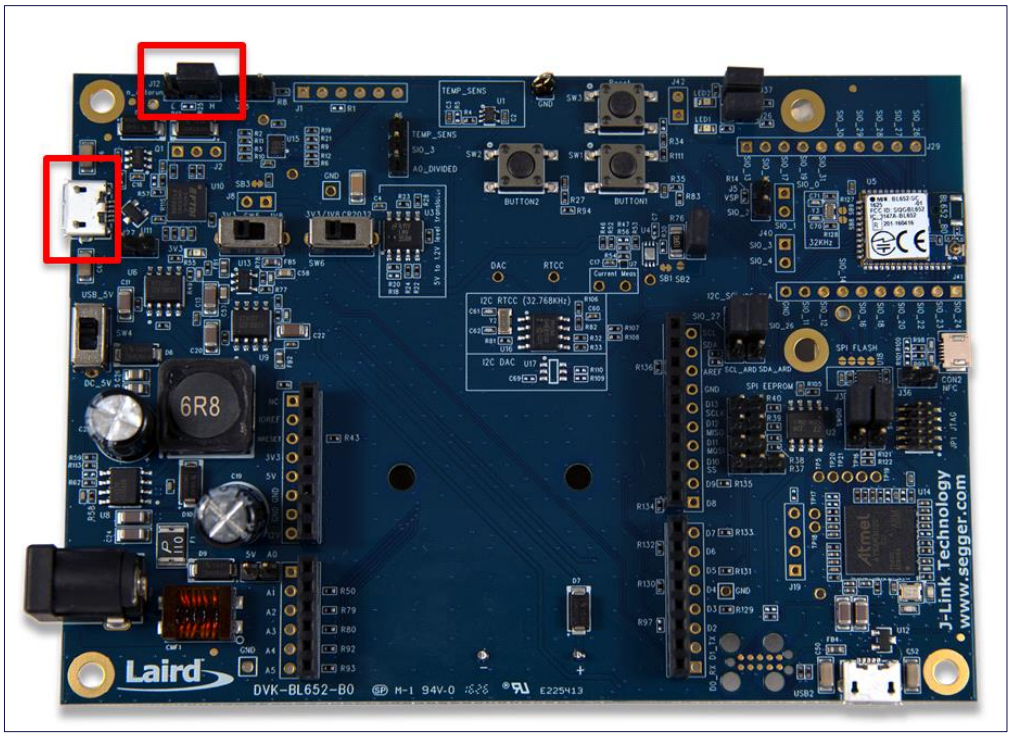


***Figure 1: USB port and jumper settings***

2. Download UwTerminalX (version 1.08n or later) from GitHub. Open two instances, and after accepting the license terms, you may select the BL652 platform from the device dropdown list.
3. In each instance of UwTerminalX's Config tab, select one of the COM ports connected to your BL652 devices. Click **OK** to connect.

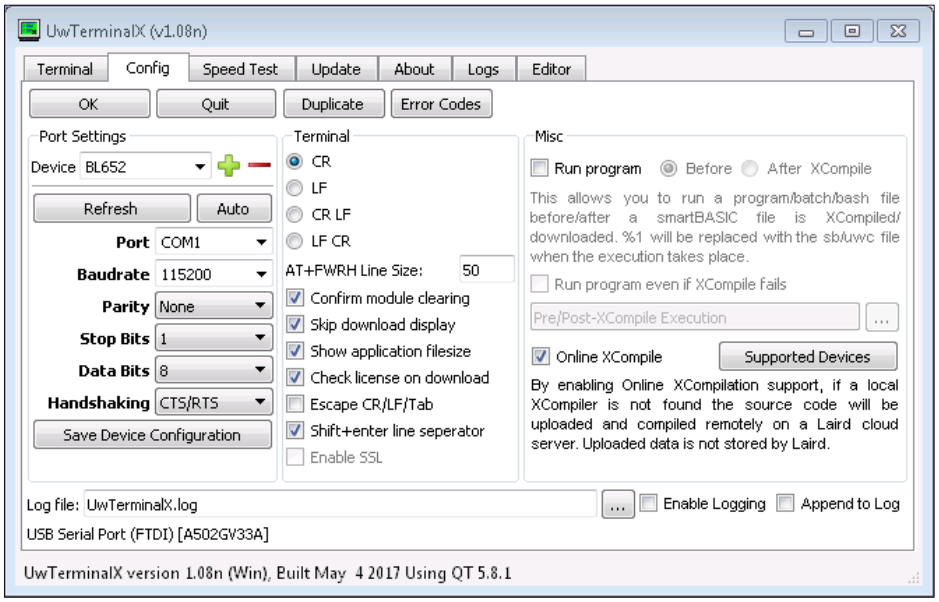


***Figure 2: UwTerminalX Config tab***

4. In the UwTerminal window for the board you will use as the peripheral device, navigate to the terminal tab. Issue the command **at I 4** to determine the device's Bluetooth address. Note the response value.

5. Go to the BL652 Applications folder in the LairdCP GitHub page and click **Clone or Download**->**Download ZIP**.
6. Extract the zip file and open the $autorun$.dle.data.length.extension.central.sb application in the Applications folder with your plain text editor of choice.
7. Replace BTAddr with the Bluetooth address of the peripheral BL652 that you noted in Step 4. This is required so that the central device auto connects to the peripheral device using the address provided.

```
23  //*****************************************************************************
24  //
25  //*****************************************************************************
26
27  //This is the target Bluetooth device to connect to, 7 bytes in hex
28  #define BTAddr "000016A4B75202"
29
```

***Figure 3: Replacing BTAddr in the central application script***

8. Uncheck DTR on both of the two open UwTerminalX instances.

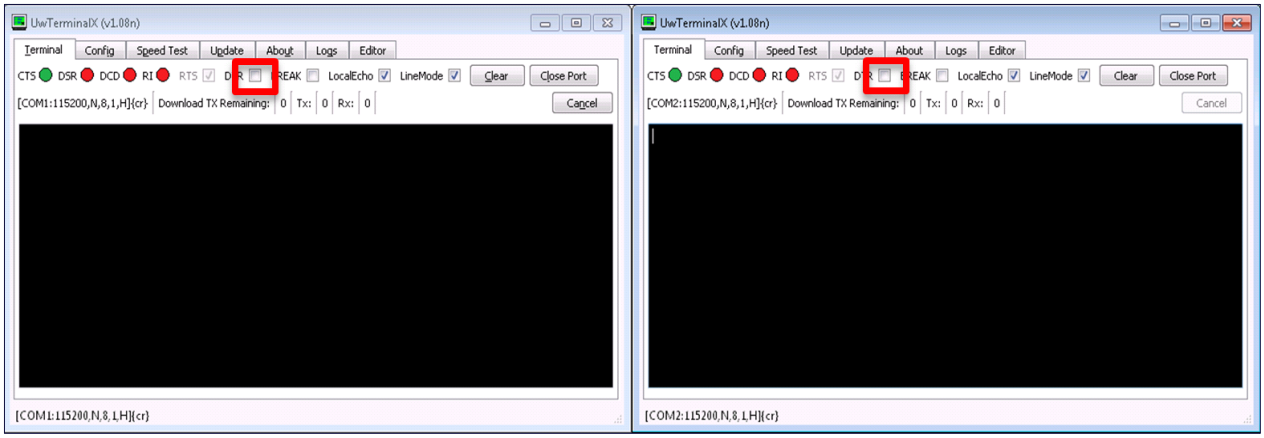


***Figure 4: Unchecking DTR in terminal windows***

9. Reset both BL652s by checking and unchecking **BREAK** on both UwTerminalX instances.
10. Ensure you have the correct firmware by issuing the **ATI 3** command (must be version 28.6.2.0 or later).
11. Flash erase both BL652s using by typing **at&F***followed by hitting enter.

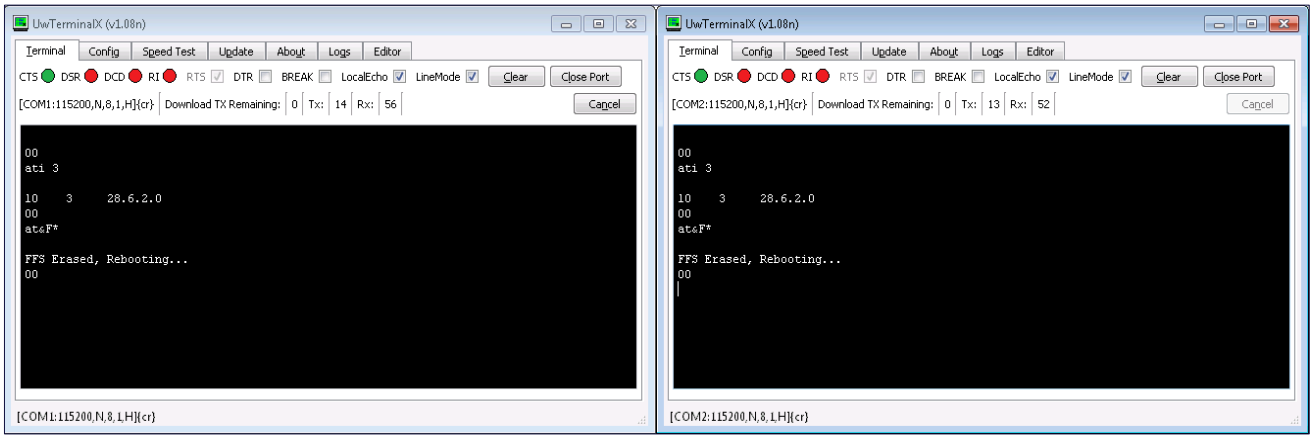


***Figure 5: Flash erasing both devices***

12. Right-click on UwTerminalX window for the device that will serve in the peripheral role (determined in step 4), and select **XCompile+Load+Run** *$autorun$.dle.data.length.extension.peripheral.sb* that was obtained in step 3.
13. Righ-click on the other UwTerminalX window and select **XCompile+Load+Run** *$autorun$.dle.data.length.extension.central.sb* that was obtained in step 3.
14. On both UwTerminalX instances switch to the **Config** tab and set the **Baudrate** to *1000000* and click **OK**.
15. Check **DTR** on both devices (this ensures that when the device is reset, the $autorun$ applications will run).
16. Reset both devices by checking and unchecking **BREAK**. The devices should automatically connect and connection messages should be displayed on both as seen below.

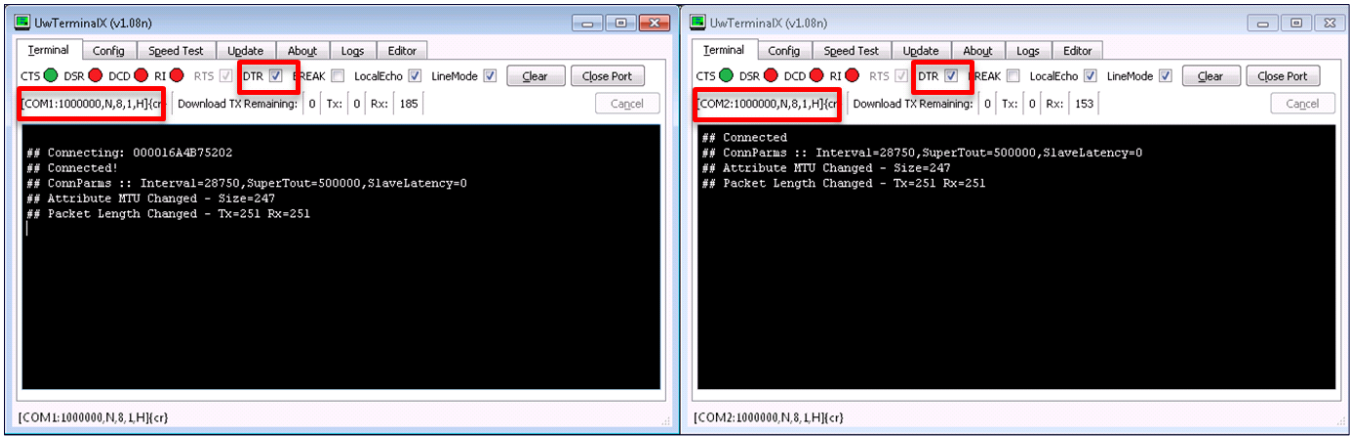


***Figure 6: DTR enabled and baud rate set correctly***

You can see that the data packet length has increased to 251 bytes and the ATT_MTU to 247 bytes.

17. Test that the devices can communicate by typing text on each terminal window and hitting 'Enter'.
18. Switch to the **Speed Test** tab on both UwTerminalX windows.
19. On first UwTerminalX window, click **Start Test**-> **Send & receive test (delay 5 seconds).**
20. On second UwTerminalX window, click **Start Test**-> **Send & receive test**.

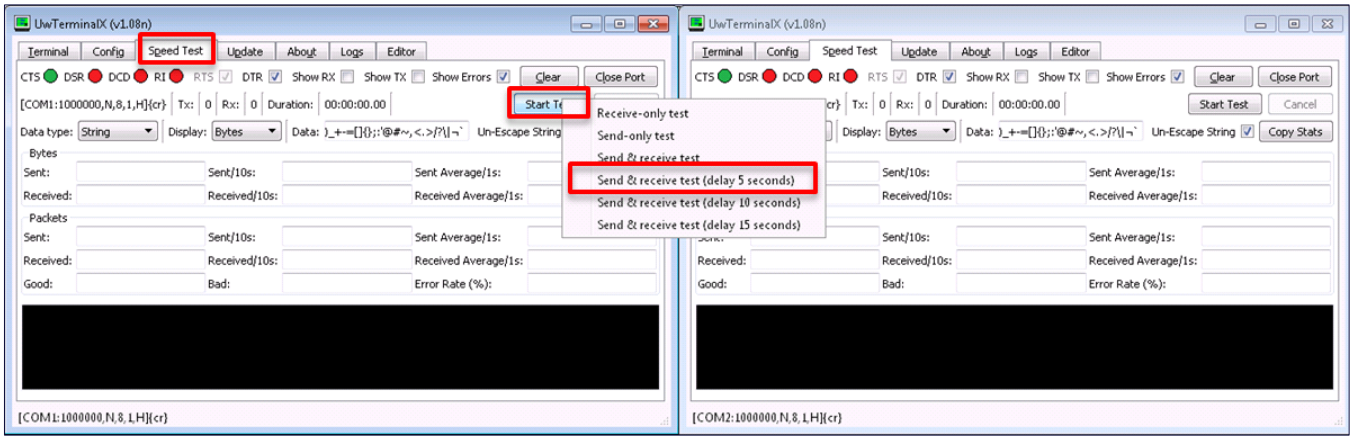


***Figure 7: Starting send and receive test***

After a few minutes of testing you can note down the throughput values in the Received Average/1s slots. You can also change the Display to show the values in Bytes, Data Bits, or Bits.

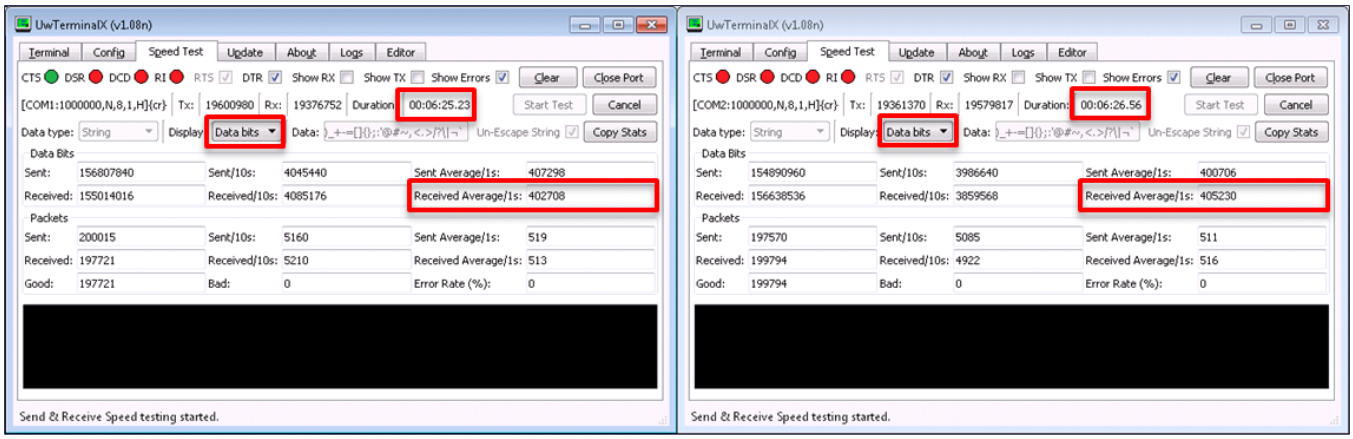


***Figure 8: Speed test results***

The results above show that the approximate throughput is 400kbps per direction, which means that the bidirectional throughput is equal to **~800kbps**.

## Test Parameters

| | |
|---|---|
| **UART Baud Rate** | 1000000 |
| **Connection Interval** | 28.750 ms |
| **Bandwidth Configuration** | Set to HIGH (6 packet per connection interval) |
| **Packet Length** | Set to 251 bytes when Data Length Extension is implemented (default is 27). |
| **Attribute MTU** | Set to 247 bytes when Data Length Extension is implemented (default is 23). |
| **Attribute Data Length** | Set to 244 bytes when Data Length Extension is implemented (default is 20). |

# LE Data Packet Length Extension Overview

One of the major additions in Bluetooth v4.2 is LE Data Packet Length Extension. This feature allows the BLE packet size to increase from 27 to 251 bytes in the link layer, therefore significantly increasing the capacity of the data channel. The benefits of this feature include:-

- **Throughput** – As seen from the steps above, the BLE throughput is significantly improved, as less time is required to transfer the same amount of data compared to previous versions of BLE.
- **Power Consumption** – Fewer transactions are required to transfer a given amount of data compared to previous versions of BLE. This reduces the time for which the radio is active, resulting in lower power consumption.
- **Fragmentation** – Fragmentation is improved as data that is larger than 27 bytes can now be sent in fewer packets. This is particularly important for the Internet of Things (IoT) market, as less fragmentation is needed for IPv6 data packets that would otherwise be split into 27 byte packets.
- **Efficiency** – Less overhead is needed for the same amount of data, therefore improving the efficiency.

In order to take full advantage of the improvements provided by the Packet Length Extension Feature, the device should also support larger ATT_MTU and bigger attribute data lengths. These are described in the sections below.

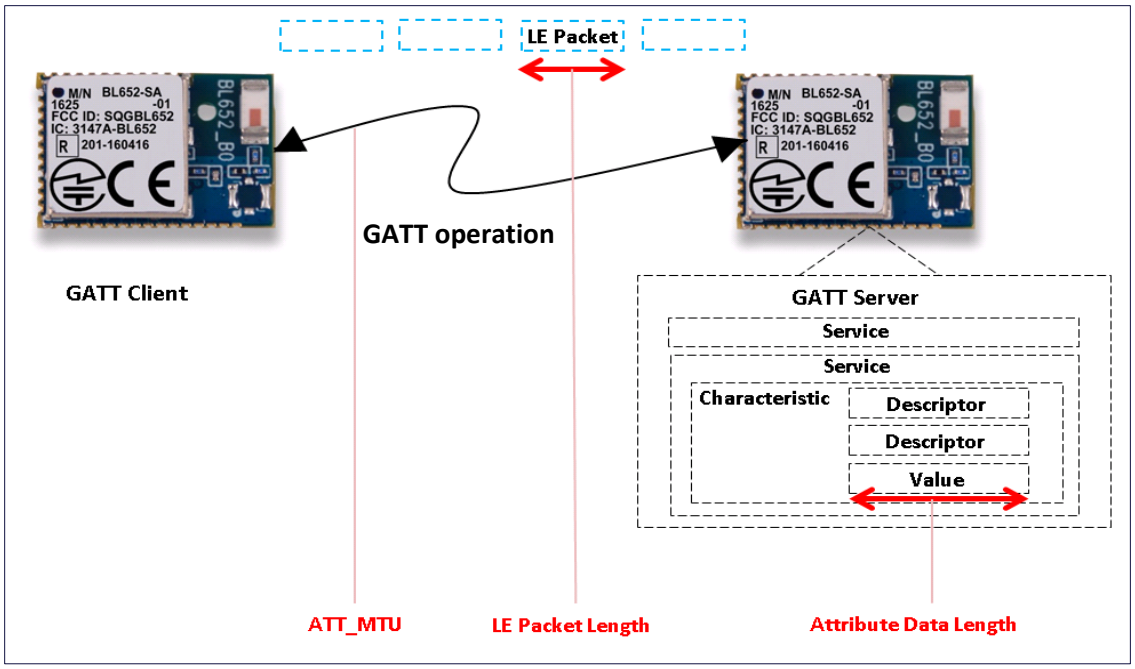


***Figure 9: LE Data Packet Length diagram***

## Attribute Data Length

The Attribute Data Length is the size of the data inside a GATT server. In order to send larger data over the air, the data residing in the GATT server should be increased as well. The default maximum attribute data size on the BL652 is 20 bytes, and can be increased to 244 bytes. This is done using **AT+CFG 212 num** before startup, or using the *smart*BASIC function **NvCfgKeySet(212, num)**. In both cases, the board should be reset (using ATZ or Reset(0) ) so that the new value takes effect. When creating a new attribute, the function **BleAttrMetaDataEx** can then be used to create characteristic values larger than 20 bytes. Please see the BL652 Extension Guide for more details, found in the documentation tab of the BL652 product page at Lairdtech.com.

## ATT_MTU

The ATTribute Maximum Transmission Unit defines the amount of data a device can send/receive per GATT operation. The default ATT_MTU on the BL652 is 23, and can be increased to 247 bytes. This is done using **AT+CFG 211 num** before startup, or using the *smart*BASIC function **NvCfgKeySet(211, num)** at runtime. In both cases, the board should be reset (using ATZ or Reset(0) ). At runtime, after a connection is established, the GATT Client should use the function **BleGattcAttributeMtuRequest()** to request that the ATT_MTU be changed to the maximum supported by both devices. In order to fully utilize the use of Packet Length Extension, the ATT_MTU should always be set to (Maximum Packet Length - 4). This means that for the maximum packet length of 251, the optimum attribute MTU to set is 247 bytes. As shown from the table below, this means that at least 244 bytes of data can be sent/received per GATT operation.

| GATT Operation | Attribute Size | Example when ATT_MTU=23 |
|---|---|---|
| Read | 0 to (ATT_MTU-1) | The GATT client can read 22 bytes from a GATT server's attribute data. |
| Write | 0 to (ATT_MTU-3) | The GATT client can write up to 20 bytes to a GATT server attribute. |
| Notification | 0 to (ATT_MTU-3) | The GATT server can send notifies of data up to 20 bytes long |
| Indications | 0 to (ATT_MTU-3) | The GATT server can send indications of data up to 20 bytes long |



Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

## Data Packet Length

The BL652 supports the increase of the data packet length from the default of 27 to the maximum value allowed by the Bluetooth spec which is 251 bytes. On the BL652, increasing the packet length is achieved through the following steps:-

1- Increase the ATT_MTU size to (Desired Packet Length - 4) using AT+CFG 211 or NvCfgKeySet(211, num) followed by a reset.
2- Increase the maximum packet length supported by the device (from 27 to 251) using AT+CFG 216, or NvCfgKeySet(216, num) followed by a reset.

```
121     // Get current ATT_MTU and maximum packet length value
122     rc = NvCfgKeyGet(211, nAttributeMTU)
123     rc = NvCfgKeyGet(216, nBleMaxPacketLength)
124
125     // Check if these are the values needed to achieve DLE
126     IF  (nAttributeMTU != 247) || (nBleMaxPacketLength != 251) THEN
127         // Change the Attribute MTU to be the maximum allowed by the BL652 (247)
128         rc = NvCfgKeySet(211, 247)      <-- Equivalent to AT+CFG 211 247 before startup
129         // Change the maximum packet length to 251 bytes
130         rc = NvCfgKeySet(216, 251)      <-- Equivalent to AT+CFG 216 251 before startup
131         // Reset the module so that the data is overwritten
132         Reset(0)
133     ENDIF
```

3- Once the connection is established, an increase to the ATT_MTU should be requested from the GATT client using BleGattcAttributeMtuRequest().

```
239         // Now that everything is setup, Requst maximum MTU
240         rc = BleGattcAttributeMtuRequest(ConnectionID)
```

The function above requests a new MTU request from the remote GATT server. In the firmware, the GATT client's MTU is compared against the GATT server's MTU (which on the BL652 are both set using AT+CFG 211), and the lower value of the two is set as the ATT_MTU. For example, when a device supports an ATT_MTU of 247, requests a change to the link's ATT_MTU from a device which supports a value of 150, the link's ATT_MTU is set to 150.

Requesting a higher MTU in turn triggers a request for a higher Data Packet Length for that connection. The length requested is always set to (ATT_MTU+4). Therefore, when the link's MTU is changed to 247 bytes, the link's packet length changes to 251 bytes given that both devices support this value.

## Higher Bandwidth Configuration

As of firmware 28.6.2.0 of the BL652, the higher bandwidth configuration feature was also added. This feature enables the BL652 to send up to 6 packets per connection interval, therefore significantly improving the throughput (the default value on the BL652 is 3 packets per connection interval). Higher bandwidth configuration can be enabled using **AT+CFG 214 1** before startup or using the *smart*BASIC function **NvCfgKeySet(214, 1)**. In both cases, the board should be reset (using ATZ or Reset(0) ) so that the new value takes effect.

---

**Note:** When the High Bandwidth feature is enabled, only the first connection will be able support up to 6 packets per connection. Subsequent connections will support the default 3 packets per connection interval.

---





Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

## REFERENCE

Further information relating to different utilities used in this app note can be found here:

- BL652 Product Page - http://www.lairdtech.com/products/bl652-ble-module

## REVISION HISTORY

| Version | Date | Notes | Approver |
|---|---|---|---|
| 1.0 | 12 May 2017 | Initial Release | Youssif M. Saeed |
| 1.1 | 23 Oct 2017 | Replaced BleMaxPacketLengthSet with CFG 216 | Youssif M. Saeed |





Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610