AT Interface smartBASIC Application
BL652/BL654

Quick Start Guide

1 INTRODUCTION

Laird’s BL652 and BL654 modules are deployed with embedded smartBASIC. smartBASIC is an event-driven programming language that does not require complex toolchains and allows the quick creation of custom applications. The language provides an easy to use API to the underlying Bluetooth, NFC and other hardware and it in turn will generate events that the developer can service through handlers written in smartBASIC.

This guide assumes that you are familiar with loading smartBASIC applications into the module. If not, please refer to the application note Loading smartBASIC Applications which is available from the BL654 product page: https://www.lairdtech.com/products/bl654-ble-thread-nfc-modules

This quick start guide explains how to perform the following:

▪ Load the AT Interface application into two devices
▪ Make a BLE Virtual Serial Port connection
▪ Send the text hello from one device to the second
▪ Send the text world in the reverse direction
▪ Disconnection

2 REQUIREMENTS

To perform the steps in this document, you need the following:

▪ Two devices which could be any one of a DVK-BL652 or DVK-BL654 development board or BL654 USB dongles.
▪ USB-A to USB-micro cable (included with development board) which is not needed if you use a BL654 USB dongle.
▪ UwTerminalX (v1.10a or later recommended), available at https://github.com/LairdCP/UwTerminalX/releases
▪ BL652 smartBASIC sample applications, available at https://github.com/LairdCP/BL652-Applications or https://github.com/LairdCP/BL654-Applications in which you will find a folder called ‘ATinterface’ which contains the application that will need to be loaded.
▪ (Recommended) The latest firmware for your module which can be found on the BL652 page: http://www.lairdtech.com/products/bl652-ble-module or BL654 page: https://www.lairdtech.com/products/bl654-ble-thread-nfc-modules under the ‘Software’ section

3 HARDWARE AND TERMINAL SETUP

To begin, you must physically connect the two devices to your PC and connect to them via UwTerminalX.

To do so, complete the following:

1. Connect the USB dongle or development board to your PC via the included USB micro cable.
2. Power your development board (SW4).
3. Launch UwTerminalX.
4. From the Update tab within the UwTerminalX pane, click Check for Updates to ensure you’re using the latest version of UwTerminalX with support for the BL652 or BL654.
5. From the Config tab in the Device drop-down menu, select **BL65X**.

**Note:** The BL65X option may not appear in the drop-down menu in older versions of UwTerminalX. In these earlier versions, you must manually enter the correct settings (115200, N, 1, 8) as shown below:

6. Select the correct port to which your development board or USB dongle is connected.
7. Click **OK** to advance to the Terminal tab and check that the CTS solid circle is green and not red.
8. Press **Enter** on your keyboard. If you see the return 00, you are successfully communicating over UART.

*Figure 1: Terminal returns 00 if connected successfully*
4 Compiling, Loading, and Running The AT Application

Once you are successfully connected to the DVK-BL652, DVK-BL654, or BL654 USB dongle over UwTerminalX, you may compile, load, and run the ATInterface application.

**Note:** For cross-compiling smartBASIC applications, UwTerminalX must use a version-specific XCompiler that matches the firmware of your module. UwTerminalX, however, provides the option for using an online XCompiler. For supported firmware versions, this means that UwTerminalX finds the correct XCompiler over the Internet so that you don’t need to maintain the XCompiler on your machine. This is enabled by default and can be disabled in the Config tab.

For this quick start guide, we use the ATInterface application in the BL652 or BL654 applications GitHub library. To compile, load, and run this application, complete the steps below.

If you are using the BL652, then the application to download is called $autorun$.AT.interface.BL652.sb and if you are using the BL654, then the application to download is called $autorun$.AT.interface.BL654.sb. For convenience, given that the behavior is the same for both module types, we will just refer to the application as the ATInterface autorun application.

1. Download the ATInterface autorun application from the appropriate applications library (https://github.com/LairdCP/BL652-Applications/tree/master/ATinterface or https://github.com/LairdCP/BL654-Applications/tree/master/ATinterface) to a directory on your PC. The application is located in the ATinterface subfolder of your local repository folder.

2. If you are not already, connect to the BL652 or BL654 in UwTerminalX.

3. Right-click in the terminal window and, in the context menu, click XCompile + Load.

4. Depending on the type of module to which you are downloading, in the file selector window, select $autorun$.AT.interface.BL652.sb or $autorun$.AT.interface.BL654.sb and click Open.

5. When the terminal displays 00, the compiler has finished successfully.

6. Type at+dir and press Enter. You should see $autorun$ in the file list.

7. Perform steps 1 to 6 again but on the second device so that you have both modules loaded with the ATInterface application.

8. To run the script, type $autorun$ and press Enter or reset the module by ticking and unticking the BREAK checkbox. The latter which results in a reset of the module will mean that on startup it will find the specially named $autorun$ application and automatically run it.

9. Press Enter in the terminal. If the terminal returns OK, the module is now running the ATInterface application and will process AT commands sent to it.

10. Type AT and press Enter, and confirm that you see the following:

```
AT
OK
```

11. Type ATI0, press Enter and confirm you see the following:

```
ATI0
BL654
OK
```

The returned module type could be the BL652 or the BL654.
12. On the second module, type **ATI4**, press **Enter**, and confirm that you see the following:

```
ATI4
01EA06E7EF3E5F
OK
```

**Note:** The string (01EA06E7EF3E5F, in the example above) is the 14-hex character Bluetooth address of the second device. Your string will be a different hex string value.

Going forward, we refer to this 14-character hex string as `<mac_address>`. For your purposes, substitute this with the appropriate values for your devices.

At this point both devices are configured to accept an incoming Virtual Serial Port (VSP) connection and are advertising.

13. To check this on the first module, type **AT+LSCN** and press **Enter**. The module scans for adverts for about ten seconds and displays any adverts received.

```
AT+LSCN
AD1: 01EA06E7EF3E5F -23 "LAIRD BL654-EF3E5F"
AD1: 01EA06E7EF3E5F -23 "LAIRD BL654-EF3E5F"
AD1: 01EA06E7EF3E5F -26 "LAIRD BL654-EF3E5F"
OK
```

01EA06E7EF3E5F and LAIRD BL654-EF3E5F are specific to this example. Your numbers will be different.

14. Enter **AT+LSCN** on the second module. The first module is also advertising similarly.

The next step is to initiate a BLE connection from the first to the second module.

**Note:** Because of the ATInterface app design, this procedure is symmetrical and works either way.

15. On the first module, enter **ATD<mac_address>**.

16. After a few seconds on the first device’s UwTerminalX screen, confirm that you see the following:

```
ATD01EA06E7EF3E5F
CONNECT 0,01EA06E7EF3E5F,15000,6000000,0
```

17. On the second device’s UwTerminalX screen, confirm that you see the following:

```
RING 01C7231370B06,U
CONNECT 0,01C7231370B06,15000,6000000,0
```

On both sides, the **CONNECT** response is as follows:

```
CONNECT 0,01EA06E7EF3E5F,15000,6000000,0
```

Where:

<table>
<thead>
<tr>
<th>01EA06E7EF3E5F</th>
<th>The address of the peer device</th>
</tr>
</thead>
<tbody>
<tr>
<td>15000</td>
<td>The connection interval</td>
</tr>
<tr>
<td>6000000</td>
<td>The link supervision timeout</td>
</tr>
<tr>
<td>0</td>
<td>The slave latency</td>
</tr>
</tbody>
</table>
18. On the first device, type **hello**, press **Enter**, and confirm you see that string appear at the second device.
19. On the second device, type **world**, press **Enter**, and confirm you see that string appear at the first device.
20. Disconnect. Again, you can initiate this from either end – it’s up to you to choose.
21. On the side you choose, ensure that the LineMode checkbox is not ticked (this is essential, otherwise the escape sequence to enter AT command mode will not be recognised as it looks for idle delays)

22. Type the ^ character four times (ensure there is at least half a second delay between each ^ character being typed).

   The following displays when it is successfully disconnected:

   ![Image](image.png)

   After a disconnection, both devices automatically return to advertising mode so that another connection can either be accepted or initiated.

5 NOTES

1. In the previous section (steps 18 and 19) where **hello<enter>** and **world<enter>** were exchanged, the messages only appeared at the remote end when you pressed Enter.

   That occurred because UwTerminalX was configured for Line Mode and not because the module behaves like that inherently.

   In UwTerminalX, if you have LineMode unticked, each key is immediately transmitted as you type them.

   ![Image](image.png)

2. When a connection is established, both ends also automatically stop advertising.

3. A complete AT Interface user guide is available from the BL652 product page:


6 CONCLUSION

Laird’s smartBASIC ATInterface application is available for use in your design. This can accelerate and simplify your development of wireless by enabling an existing system that sends serial data over a wired connection. You are free and encouraged to modify the source of that application to suit your needs.

UwTerminalX, sample BL652/BL654 applications, and more are available from the following links:

- https://github.com/LairdCP/UwTerminalX
- https://github.com/LairdCP/BL652-Applications
- https://github.com/LairdCP/BL654-Applications
7 REVISION HISTORY

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Notes</th>
<th>Contributor(s)</th>
<th>Approver</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>20 June 2018</td>
<td>Initial Release</td>
<td>Youssif M. Saeed</td>
<td>Jonathan Kaye</td>
</tr>
</tbody>
</table>

Copyright 2018 Laird. All Rights Reserved. Patent pending. Any information furnished by Laird and its agents is believed to be accurate and reliable. All specifications are subject to change without notice. Responsibility for the use and application of Laird materials or products rests with the end user since Laird and its agents cannot be aware of all potential uses. Laird makes no warranties as to non-infringement nor as to the fitness, merchantability, or sustainability of any Laird materials or products for any specific or general uses. Laird, Laird Technologies, Inc., or any of its affiliates or agents shall not be liable for incidental or consequential damages of any kind. All Laird products are sold pursuant to the Laird Terms and Conditions of Sale in effect from time to time, a copy of which will be furnished upon request. When used as a tradename herein, Laird means Laird PLC or one or more subsidiaries of Laird PLC. Laird™, Laird Technologies™, corresponding logos, and other marks are trademarks or registered trademarks of Laird. Other marks may be the property of third parties. Nothing herein provides a license under any Laird or any third party intellectual property right.