Using UART Efficiently to Extend Battery Life

BL65x Series

Application Note *v2.0*

1 Introduction

This guide demonstrates how to load and run the *smart*BASIC sample application **uclp.uart.low.power.operation.sb** on the BL652 development board (DVK-BL652). This sample application shows how to close the UART when there is no UART activity and how to enable the host to reopen UART by sending a return character to make efficient use of power.

Note: This application note uses the BL652, but this can be done on any of the BL65x series modules ***except the BL651*** which has limited flash and RAM and does not support smartBASIC.

UART peripherals are, by their nature, not power efficient. To obtain optimal overall power consumption, the UART should be closed when running a *smart*BASIC application if it is unnecessary for the application. In applications where UART is necessary, you can close it when not in use to conserve power.

In this sample application, once the UART is open, it operates normally and then, if there is inactivity (as determined by a timer which is restarted when there is incoming UART activity), it is closed. If a carriage return arrives through the UART from the host after a time specified by an idle open timer, the UART reopens.

2 Overview

When the UART Rx and Tx buffers are empty, a timer starts. When it expires, the UART is closed.

```
//============================================================================
 // Called when the tx and rx buffers are empty
 //============================================================================
function HandlerUartTxEmpty() as integer
   if UartInfo(6) == 0 then
     //Start the uart inactivity timer
     TimerStart(UART_IDLE_TIMER,UART_IDLE_TIMEOUT_MS,0)
   else
     //buffers are not empty
     TimerCancel(UART_IDLE_TIMER)
   endif
 endfunc 1
```

```
//=============================================================================
 // Uart Inactivity timer handler
 //=============================================================================
 function handlerUartTimer() as integer
   dim rc
   //Close the uart, and set up TX/RX/RTS lines as gpio and for a hi-lo transition
   //on the RX line to be detected
   if UartCloseEx(1) == 0 then
     rc=GpioSetFunc(6,2,1)    '//TX  - set high on default
     rc=GpioSetFunc(5,2,0)    '//RTS - set low by default
     rc=GpioSetFunc(8,1,2)    '//RX  - Pull high input & irq on hi2lo transition
     rc=GpioAssignEvent(UART_GPIO_ASSIGN_CHANNEL,8,1)
     if rc != 0 then
       print "\nGpioAssignEvent() Failed"
     endif
   endif
 endfunc 1
```

As seen in the previous sample, an event is assigned to a high low transition on the UART Rx pin to detect when a character arrives from the host. In this event, a delay timer is started. When it expires, the UART is reopened and an acknowledgement character (**!**) is sent back to the host.

```
//=============================================================================
// Uart needs to be opened, because a hi to lo transition on RX has been detected
 //=============================================================================
 function handlerUartDetect() as integer
   //Start delay before opening
   TimerStart(UART_IDLE_OPEN_TIMER,UART_OPEN_DELAY_MS,0)
endfunc 1
```

```
//=============================================================================
// Delay before uart is opened
 //=============================================================================
 function handlerOpenDelay() as integer
   dim rc
   // free up the level transition detection
   rc=GpioUnAssignEvent(UART_GPIO_ASSIGN_CHANNEL)
   //Open the uart
   rc=UartOpen(115200,0,0,"CN81H")
   //send an ack character
   print "!"
endfunc 1
```

The following screenshot shows what the BL65x receives if you send a carriage return after the UART is closed. The BL65x acknowledges receiving the carriage return by printing ***!*** and opening the UART and data can now be sent and received.

```
!hello

 Got :hello
```

Note: The timer intervals are #defined in the sb file.

3 Requirements

The following is required for this process:

- PC running Windows XP or later
- UwTerminalX v1.10a or later
- DVK-BL652 Development Kit loaded with at v28.6.1.2 firmware or later**
- **uclp.uart.low.power.operation.sb** *smart*BASIC sample application. Available on our GitHub for the BL652: https://github.com/LairdCP/BL652-Applications
  (If working with BL653, BL654, or newer module please access the Applications repository for that module from GitHub)
- USB A to micro USB cable
- DVK_BL65x User Manual
- FTDI Drivers http://www.ftdichip.com/Drivers/VCP.htm (for some versions of Windows)

** The latest BL652 firmware and upgrade documentation is available at the following link

https://www.ezurio.com/wireless-modules/bluetooth-modules/bluetooth-5-modules/bl652-series-bluetooth-v5-nfc-module under the Software tab.

(See Product Page of other BL65x modules for the appropriate documentation.)

4 Development Kit Setup

To setup the BL65x development kit, follow these steps:

1. Configure the BL65x development kit to the following settings (Figure 1):
   - 1: DC/USB power source switch (SW4) – USB
   - 2: VCC_1V8/VCC_3V3 switch (SW5) – VCC_3V3
   - 3: CR2033/VCC_3V3/1V8 switch (SW6) – VCC_3V3/1V8
   - 4: Jumper J6 (TEMP_SENS) off
   - 5: Jumpers J26 and J37 (LED1 and LED2) off
   - 6: Jumper J7 (Current Meas) off
   - 7: Jumper J5 (VSP) off

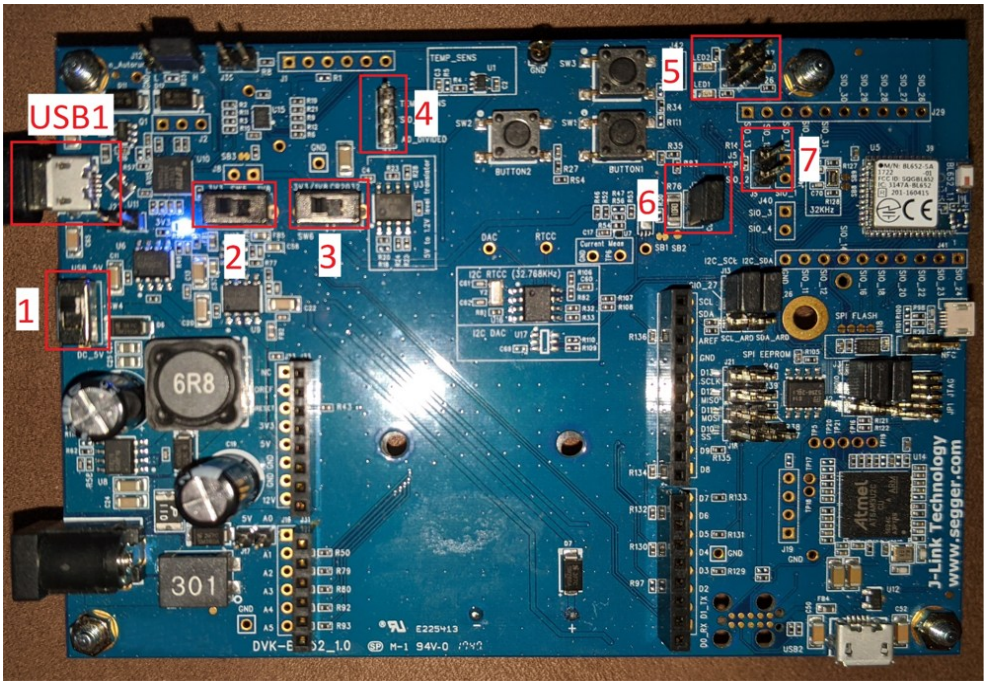

Figure 1: Switch and jumper positions for the BL652

2. Connect the micro USB cable to USB1 on the development board and connect the other end of the cable to your PC.
3. Follow the on-screen prompts. Depending on your version of Windows, you may need to install the FTDI drivers.

   When complete, the development board appears in the Windows device manager as a *USB Serial Port*.
4. Open UwTerminalX.
5. Configure the COM port with the port number seen in the device manager with the following settings (Figure 2):
   - Baudrate – 115200
   - Stop bits – 1
   - Data bits – 8
   - Handshaking – None
6. Confirm that you can communicate with the development board by typing *at* followed by a carriage return. The module should respond with *00* (Figure 2).

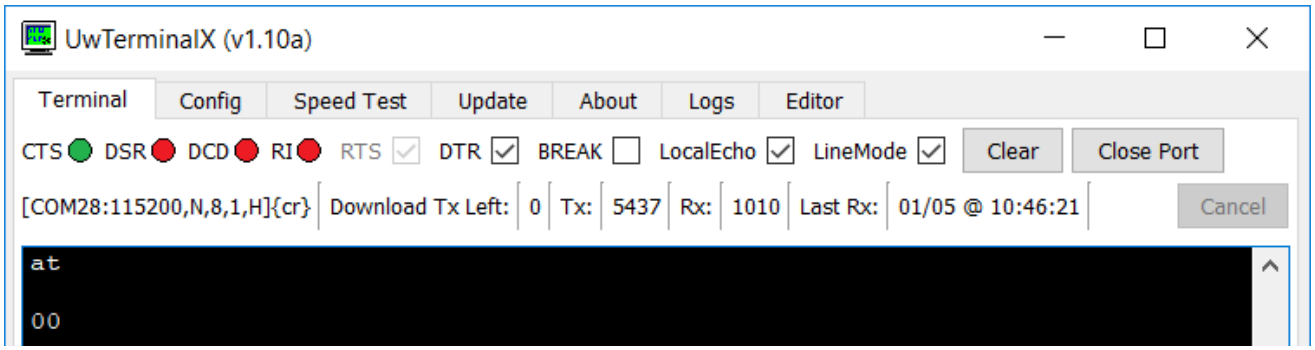

Figure 2: Communications OK

5 Loading and Running the smartBASIC Application

To load a *smart*BASIC application, follow these steps:

1. To compile and load a *smart*BASIC application, right-click in the main UwTerminalX window and select **XCompile + Load** (Figure 3).

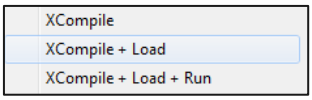

Figure 3: Right-click menu

2. Locate and open the **uclp.uart.low.power.operation.sb** application located in the supplied *smart*BASIC ***Applications*** folder downloaded from GitHub. When the application is successfully compiled and loaded, the console displays **-- Finished downloading file --** (Figure 4).

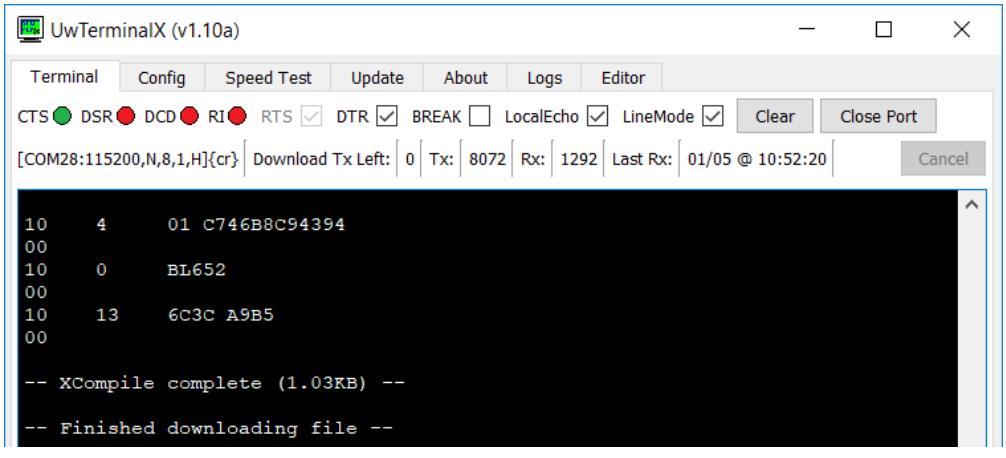

Figure 4: Application compiled and loaded

3. Confirm that the **uclp** application is loaded by using the command **at+dir** (Figure 5).

Note: The file extension is truncated from files copied onto the BL65x modules. Therefore, when uclp.uart.low.power.operation.sb is copied to the device, its name becomes *uclp*.

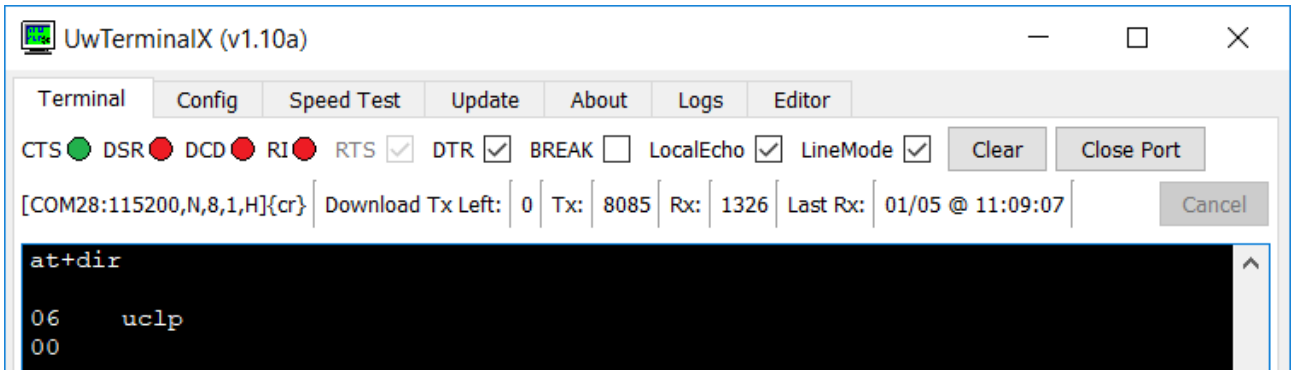

Figure 5: Directory showing "uclp" app is loaded

4. To run the application, type *uclp* and press the return key.
5. To see the change in power when UART is closed compared to when it is open, remove the J7 jumper and attach a multimeter to enable current measurement. J7 is located at ***4*** on Figure 1. Ensure the CTS light is green while connected to the multimeter and check the connections if not.
6. The UART closes after four seconds passes with no UART activity. Press Enter to open the UART, as described in the *Overview* section. The multimeter should show a higher current (Figure 6) when the UART is open and a lower current (Figure 7) when the UART is closed.

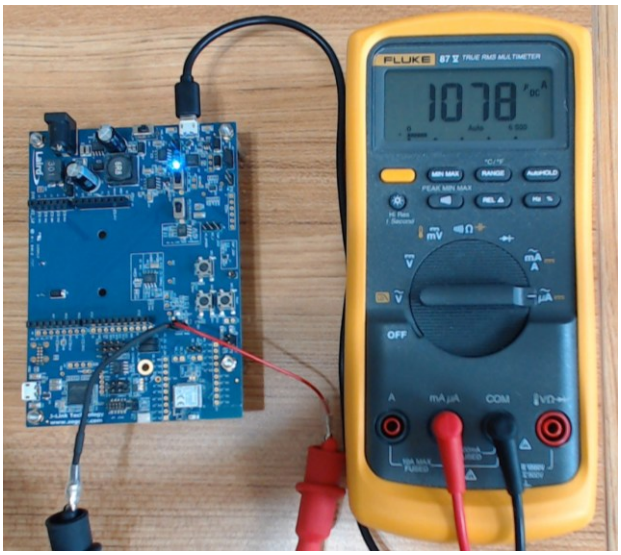

Figure 6: UART open

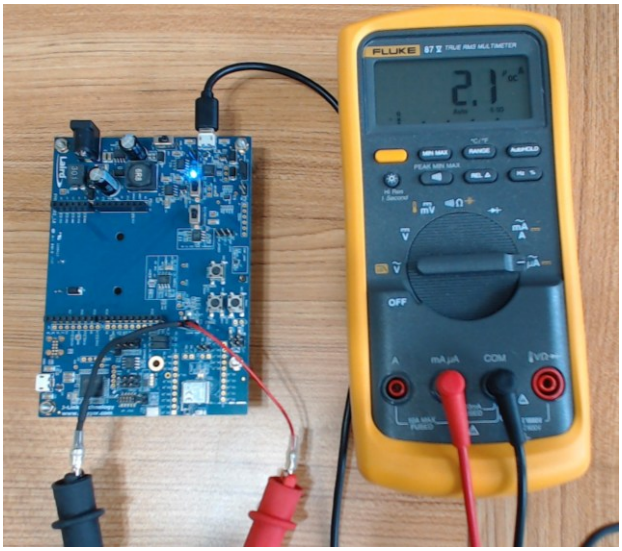

Figure 7: UART closed

6 References

For more information on the UART as well as any *smart*BASIC commands used in this application note, refer to the smartBASIC Module user guide which can be accessed from the from any of the BL65x product pages on the Ezurio website.

Additional product information can also be accessed from the same product pages:

BL652: https://www.ezurio.com/wireless-modules/bluetooth-modules/bluetooth-5-modules/bl652-series-bluetooth-v5-nfc-module

BL653: https://www.ezurio.com/wireless-modules/bluetooth-modules/bluetooth-5-modules/bl653-series-bluetooth-51-802154-nfc-module

BL654: https://www.ezurio.com/wireless-modules/bluetooth-modules/bluetooth-5-modules/bl654-series-bluetooth-module-nfc

BL654PA: https://www.ezurio.com/wireless-modules/bluetooth-modules/bluetooth-5-modules/bl654pa-series-long-range-bluetooth-module

7 Revision History

Version	Date	Notes	Contributor(s)	Approver
1.0	22 May 2020	Initial Release	Gina Lanese Rikki Horrigan	Jonathan Kaye
2.0	27 Feb 2025	Converted to Ezurio branding	Sue White	Dave Drogowski