USING UART EFFICIENTLY TO EXTEND BATTERY LIFE

Application Note *v1.1*

INTRODUCTION

This guide demonstrates how to load and run the *smart*BASIC sample application uclp.uart.low.power.operation.sb on the BL600 development board (DVK-BL600). 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 sacrificial character to make efficient use of power.

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 your application.

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 character arrives through the UART from the host after a time specified by an idle open timer, the UART is reopened.

*SMART*BASIC APPLICATION 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(21,2,1)    '//TX  - set high on default
    rc=GpioSetFunc(23,2,0)    '//RTS - set low by default
    rc=GpioSetFunc(22,1,2)    '//RX  - Pull high input & irq on hi2lo transition
    rc=GpioAssignEvent(UART_GPIO_ASSIGN_CHANNEL,22,1)
    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(9600,0,0,"CN81H")
  '//send an ack character
  print "!"
endfunc 1
```

The following screenshot shows what the BL600 receives if you type *hello* followed by a carriage return after the UART is closed. The BL600 acknowledges receiving the ***h*** character by printing ***!***, opens the UART, and prints the remaining data read from the Rx buffer.

```
h!ello

 Got :ello
```

Note: The timer intervals are #defined on lines 28 and 29 of the sb file.

REQUIREMENTS

- PC running Windows XP or later
- UWTerminal 6.50 or later
- DVK-BL600 Development Kit loaded with at v1.2.54.0 firmware or later **
- **uclp.uart.low.power.operation.sb** *smart*BASIC sample application
- USB A to mini B cable
- DVK_BL600 User Manual
- FTDI Drivers http://www.ftdichip.com/Drivers/VCP.htm (for some versions of Windows)

** The latest BL00 firmware and upgrade documentation is available at the following link:
https://laird-ews-support.desk.com/?b_id=1945#docs

Note: Some documentation requires access to the BT Firmware Download Center. Click here to request access.

Product information can also be accessed from the BL600 product page on the Laird website:
http://www.lairdtech.com/products/bl600-series

DEVELOPMENT KIT SETUP

To setup the BL600 development kit, follow these steps:

1. Configure the BL600 development kit to the following settings (Figure 2):
   - DC/USB power source switch (SW4) – USB
   - VCC_1V8/VCC_3V3 switch (SW5) – VCC_3V3
   - CR2033/VCC_3V3/1V8 switch (SW6) – VCC_3V3/1V8

Figure 2: Switch and jumpers position

2. Connect one end of the mini USB cable to CON4 on the development board and 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. Extract UWTerminal to a selected folder and run the program.
5. Configure the COM port with the port number seen in the device manager with the following settings (Figure 3):
   - Baudrate – 9600
   - Stop Bits – 1
   - Data Bits – 8
   - Handshaking – None

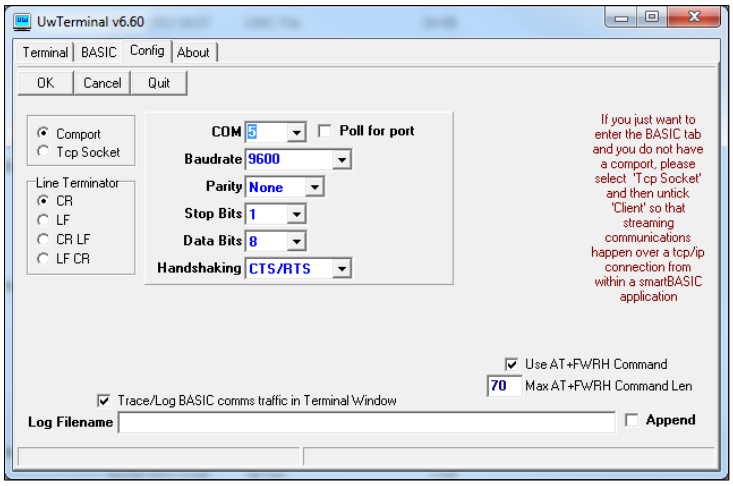

Figure 3: Comms Settings

6. Confirm that you can communicate with the development board by typing *at* followed by a return. The module should respond with *00*. (Figure 4)

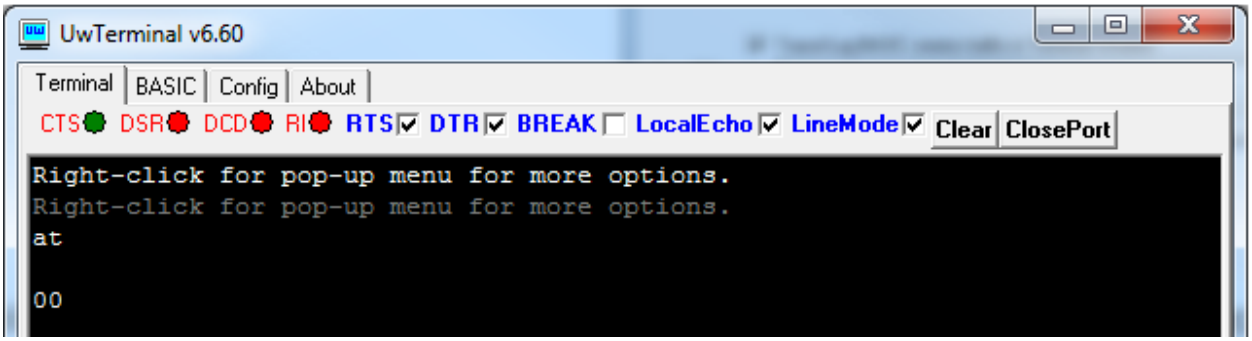

Figure 4: Comms OK

LOADING THE *SMART*BASIC APPLICATION

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

1. Ensure the cross compiler is located in the same folder as UWTerminal. Its name is similar to XComp_BL600r2_CA0D_1DA6, where *CA0D_1DA6* indicates a hash key. Each firmware version requires its corresponding cross compiler with a matching hash key.
2. To compile and load a *smart*BASIC application, right-click in the main UWTerminal window and select **XCompile + Load** (Figure 5).

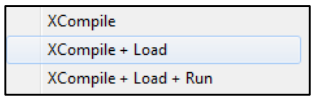

Figure 5: Right-click menu

3. Locate and open the **uclp.uart.low.power.operation.sb** application located in the supplied *smart*BASIC *_sample_Apps* folder. When the application is successfully compiled and loaded, the console displays **+++ DONE +++** (Figure 6).

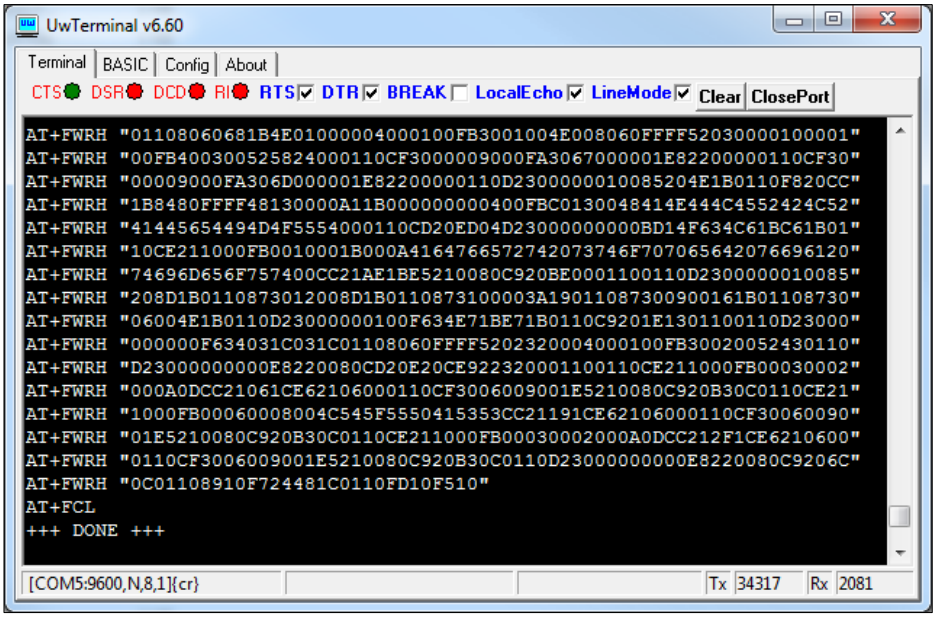

Figure 6: Compiled and loaded

If the correct version of cross compiler is not present, an error displays.

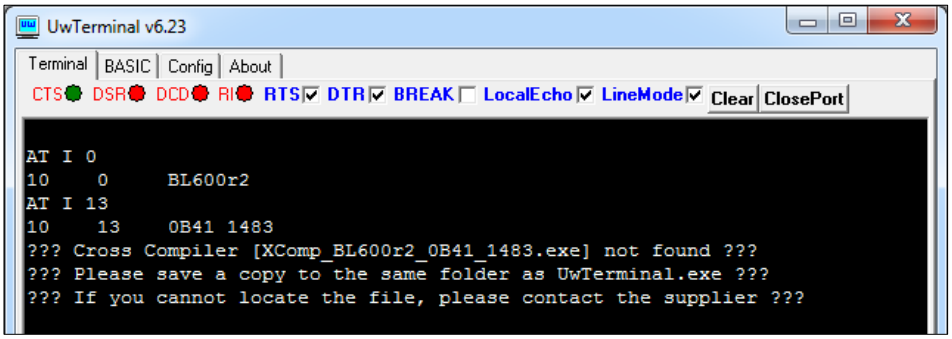

Figure 1: Cross compiler error

4. Locate the correct version and place it in the same folder as UWTerminal.
5. Confirm that the **uclp** application is loaded by using the command **at+dir** (Figure 2).

Note: The file extension is truncated from files copied onto the BL600 module. Therefore, when **uclp.uart.low.power.operation.sb** is copied to the device, its name becomes ***uclp***.

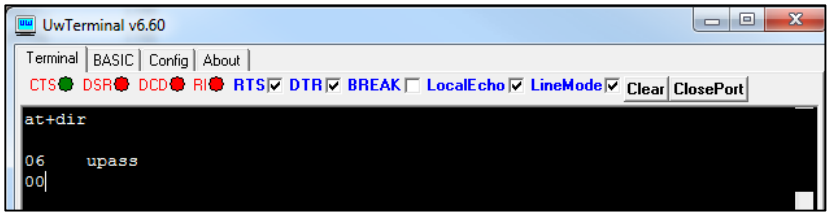

Figure 2: Directory showing "uclp" app loaded

REFERENCES

For more information on the UART as well as any *smart*BASIC commands used in this application note, refer to the BL600 smartBASIC Module user guide which can be accessed from the Embedded Wireless Solutions Support Center: https://laird-ews-support.desk.com/?b_id=1945#docs

Product information can also be accessed from the BL600 product page on the Laird website: http://www.lairdtech.com/products/bl600-series

REVISION HISTORY

Revision	Date	Description	Approved By
1.0	10 Nov 2014	Initial Release	Jonathan Kaye
1.1	08 Jan 2015	Updated *References* links to new website	Sue White