Manual

BC Mesh

BC Mesh 1.0.0 Manual Rev. A

Features

- Configurable Bluetooth Low Energy (BLE) software
- Provides flexible Mesh environment, allowing messages to be sent to a specific node or broadcasted to all nodes
- Can be controlled over UART or via GPIO interfaces
- Controllable over the air by: Android 4.3+ BLE devices, iOS support pending
- Highly flexible and configurable interface

Applications

- BLE sensors/telemetry
- BLE Remote control and automation
- BLE data transfer

Description

BC Mesh is an embedded firmware solution running on the CSR10XX family chips. It allows the user to build and control a Mesh network from their mobile device.

It connects to Android 4.3+ BLE enabled devices. It is highly customisable and designed to enable fast development of BLE based products

Introduction

BC Mesh is an embedded firmware solution running on the CSR10XX family chips. It allows the user to build and control a Mesh network from their mobile device.

A mesh network is an ad-hoc Bluetooth network, which allows data transfer to any of the devices associated with the network. Each BC Mesh device acts as a node in such a network. The network topology is controlled by a mobile device, such as an Android smart phone.

BC Mesh includes a control application and the Mesh-enabled low level firmware. It therefore allows implementing a BLE Mesh network without any detailed knowledge of the Bluetooth standard. The BC Mesh firmware allows the host processor to control and oversee the data transfer through simple UART commands.

Setting Up

To start you need to have:

a) BC118 Discovery Board or CSR101x Development Board.
   - Please contact sales@bluecreation.com for more information.

b) A computer running a serial terminal, such as PuTTY, HyperTerminal for Windows or an equivalent program, to communicate over the COM interface.

The CNS10020 board enumerates as a virtual COM port. Please use Device Manager to discover the port

number. By default, BC Mesh uses the following UART settings:

- Baud rate : 9600bps
- Data bits : 8
- Stop bits : 1
- Parity bit : No parity
- RTS/CTS Flow Control : Disabled

A picture of a development board is depicted bellow

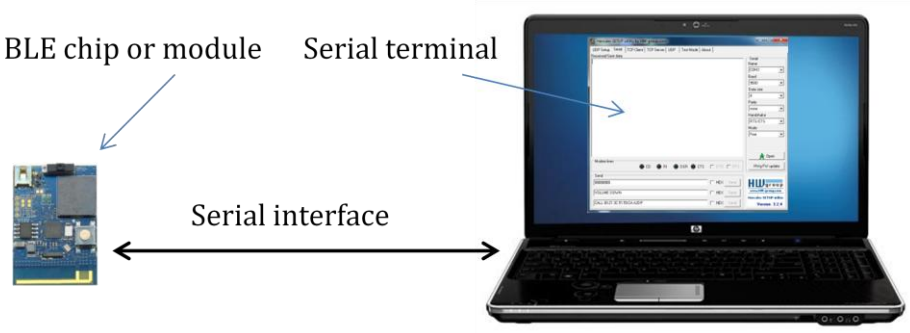


Figure 1: Example configuration

Once you have configured your serial terminal and opened the COM port, power up the development kit. You should see a prompt appear on the screen of the terminal. If you see a prompt and a READY, the module is ready to operate.

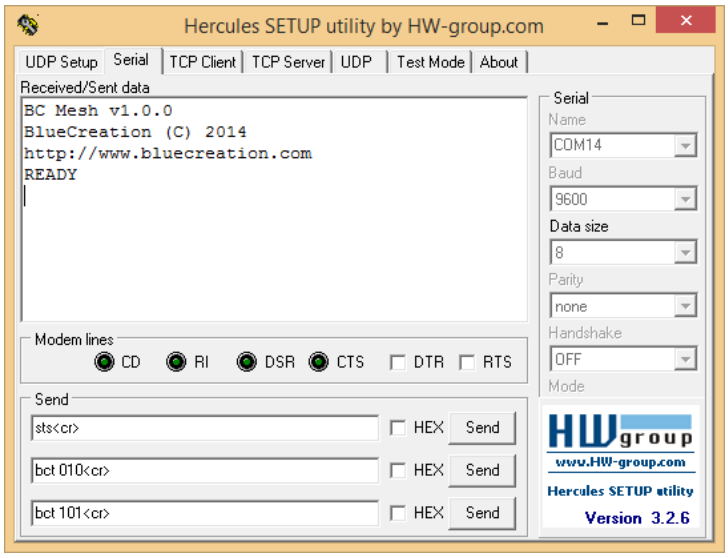


Figure 2: BC Mesh boot-up prompt

You are now ready to control the chip!

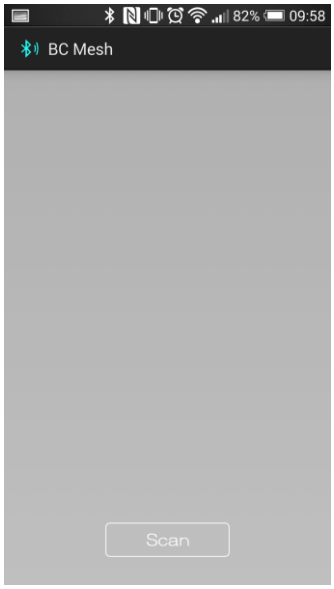
If you do not see the prompt and READY appear, please check:

1) That the development is powered ON and receiving power
2) The UART settings on your serial terminal are the same as above
3) The FTDI Drivers are installed correctly
4) Check with your distributor that your module is loaded with BC Mesh

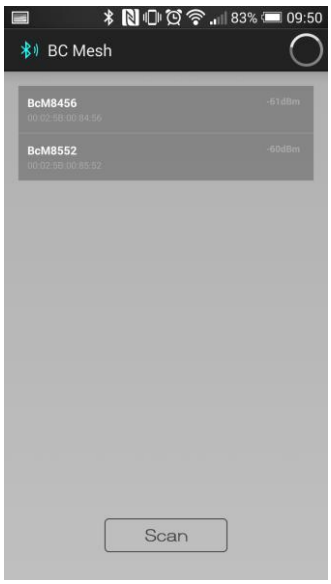
Example Workflow

The creation of a network consists of the following steps:

1. Open the BC Smart Android application.

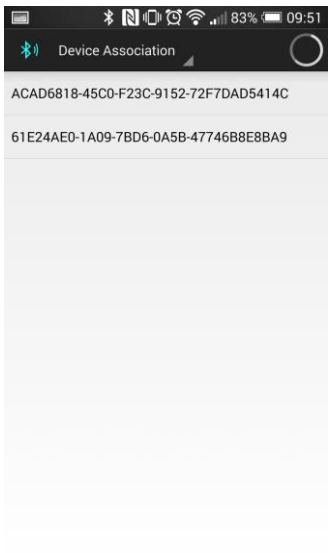


2. Press "Scan" in order to find the available BC Mesh-enabled devices.



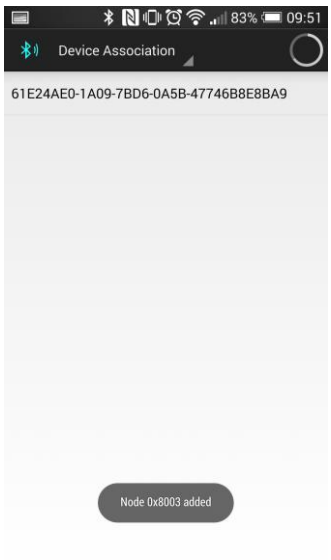
3. Connect to one of the displayed devices by tapping on it.

4. A list of UUIDs of all currently unassociated devices is presented to the user.



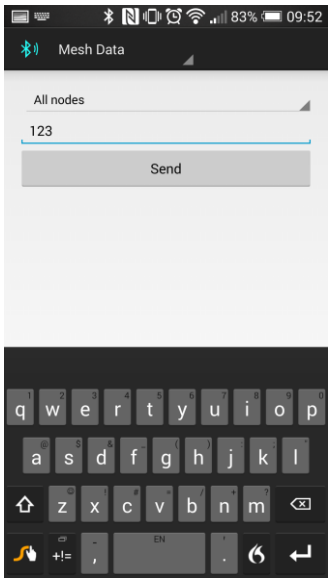
5. By selecting an UUID from the list, the device gets associated and added to the current Mesh network.

6. After a successful association, each BC Mesh node is given a 16-bit numeric Device ID, which is shown in a Pop-up notification.

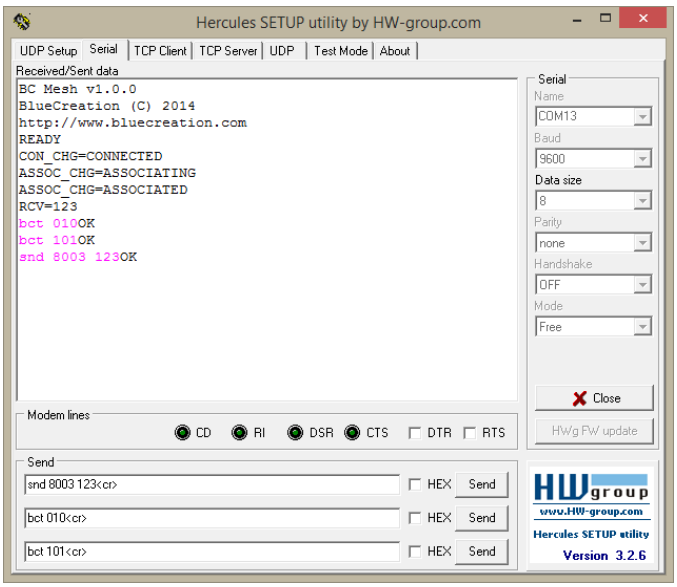


7. When all desired nodes have been associated with the network, the user can switch to the "Mesh Data" tab.

8. The "Mesh Data" tab allows the user to broadcast 3-byte messages to any or all one of the nodes in the network.



9. Each node reports the received data in a "RCV=XXX" prompt.



10. You can also send data between the nodes by using the BCT and SND commands. For example, "BCT 123" broadcasts the 3 bytes "123" to all nodes in the network.

UART Commands

When in Command mode, the module accepts commands from the host via the UART interface. The generic syntax for commands is:

`COMMAND (parameter_1) (parameter_2) … (parameter_n)\r`

with a space between each parameter and a carriage return (‘**`\r`**’ or **`0x0D`**) at the end of each command.

BC Smart will return an **`‘OK’`** after each command to indicate that the command was executed. An **`‘ERR’`** will be returned if the command has not been executed or if the parameters are wrong.

The different commands to control the Bluetooth link are listed in alphabetical order below. Mandatory parameters are listed in "**`( )`**".

The different UART Commands to control the Bluetooth link are listed in alphabetical order below.

UART Command	Description
`BCT (data)`	Broadcasts `data` to the Mesh. Note that `data` can currently be exactly 3 bytes long. Data cannot contain ‘\r’ (ASCII 0x0D) as this will be interpreted as the end of the command.
`CFG`	Displays all current configurations. See Table 2 for details.
`GET (config)`	Gets the current setting for configuration/s. See Table 2 for details.
`HLP`	Displays available commands.
`RST`	Resets the chip

SET (config)=(params)	Sets parameters for specified configuration. See Table 2 for details.
SND (address) (data)	Sends `data` to the given Mesh node address. Note that `data` can currently be exactly 3 bytes long. Data cannot contain '\r' (ASCII 0x0D) as this will be interpreted as the end of the command. The address contests of 4 hexadecimal digits.
STS	Shows the device status in following format: STS =(CONN_STATE) (ASSOS_STATE) CONN_STATE can be INIT/FAST_ADV/SLOW_ADV/CONN/DISCONN/IDLE for Initializing, fast advertisement, slow advertisement, connected, disconnecting, idle. ASSOS_STATE can be NOT_ASSOCIATED/ASSOCIATING/ASSOCIATED.
UNA	Unassociates from the current Mesh
VER	Shows BC Mesh version
WRT	Write all configuration parameters to NV memory. See Table 2 for details.

Table 1: UART commands

UART Configuration Commands and Parameters

When BC Mesh is in Command mode, the user can configure general parameters for the module. These parameters are stored in the RAM memory. If required, the parameters can be stored to NV memory. When the module reboots, it will boot with the parameters that are saved to NV memory.

The commands to modify and read configuration parameters are described below. Mandatory parameters are listed in "**( )**" .

All numerical values are integers and must be supplied in ASCII hexadecimal representation.

Command	Description
CFG	Displays all current configurations and their settings. Note that if a configuration has been changed it will be shown here even if it is not stored.
GET (config)	Displays all current configurations and their settings or a specific configuration if specified. Note that if a configuration has been changed it will be shown here even if it is not stored.
SET (config)=(params)	Sets a configuration with the supplied parameter or parameters. Please refer to Table 3 for details on available configurations and settings.

	Note that all configurations require WRT in order to be stored permanently; most require a RST to be applied.
WRT	Stores all configurations. These are then loaded at boot time.

Table 2: UART configuration commands

The different configuration parameters are described in alphabetical order in the table below. Once modified, some configuration parameters will not take effect before a reboot. Therefore they need to be stored to NV memory before rebooting.

Configurations & Parameters	Description	Default	Requires reboot
ADDR	Get the BT address of the module. Read-only.	**N/A**	NO
FCTL=(state)	**state** can be ON or OFF. On enables XON/XOFF flow control for UART. Recommended ON to ensure no data loss.	**OFF**	YES
GPIO=(state)	**state** can be ON or OFF. On enables GPIO control and indications.	**OFF**	NO
LEDS=(state)	**state** can be ON or OFF. On enables LED indications on incoming data. The 0th bit of each of the 3 received bytes controls one of the LEDs.	**OFF**	NO
NAME=(name)	Device full name. Maximum of 27 characters. No terminating "\0" required. This can be read by a connected central. By default this is BcM followed by the last 5 characters of the device BT address.	**BcMXXXXX**	NO
UART=(BAUD)	Speed can be: 2K4 = 0x000a, 9K6 = 0x0028, 19K2 = 0x004e, 38K4 = 0x009e, 57K6 = 0x00eb, 115K2 = 0x01d9. Takes hexadecimal value in format XXXX where X is a hexadecimal digit. All UART communication is done with no RTS/CTS, no parity, 1 stop bit.	**0028**	YES
UUID	Returns the generated UUID for this Mesh node. It is used in associating with the device. Read-only.	**N/A**	NO

Table 3: UART configurations and parameters

Note all parameters require a write to be stored. The ones listed as not requiring a reset are applied in real time but will be lost if the device is power cycled without a `WRT` command beforehand.

UART connections

Signal	PIO	Direction
Tx	0	Output
Rx	1	Input

Table 4: 2- and 4- wire UART PIO connections

GPIO Control and signals

Some of the module's functions can also be controlled by PIOs. This allows the module to work autonomously without the need of a host processor. The table below lists the UART commands that also have a PIO equivalent. This PIO functionality can be turned on or off (See configuration parameters).

GPIO[1]	UART Equivalent	Description	Direction
3	**UNA**	Disassociate from the current Mesh.	Input
2	–	Indicates the association state of the current device. Associated = 1, Not Associated = 0	Output

Table 5: GPIO Control and signals

LED State indication

For demonstration scenarios, the module can have LED indications enabled on GPIOs 4, 9 and 10. LEDs are used to visually indicate the reception of data over the Mesh. Each LED is controlled by the rightmost bit of the corresponding received byte. As an example, a reception of the ASCII sequence '010' will turn the Red and Blue LEDs OFF, and the Green LED ON.

PIO	LED schematic name	LED Colour
10	LD5	Red
9	LD4	Green
4	LD3	Blue

Table 6: LED behaviour

BC Mesh Prompts

BC Mesh uses ‘prompts’ to notify the host of events in the Bluetooth link (access requests, connection, pairing

[1] Refer to BC111 Datasheet for location of PIO on the module PINOUT

information, etc) , the chip's PIOs and interfaces (change in PIO states or external interrupts), to provide information, or require action.

The range and availability of prompts is fully customizable based on customer specifications.

Prompt	Description
ERR	Error when a command has not been executed or the parameters are not correct.
STS=(CONN_STATE**)** **(**ASSOS_STATE**)**	CONN_STATE can be: INIT/FAST_ADV/SLOW_ADV/CONN/DISCONN/IDLE for Initializing, fast advertisement, slow advertisement, connected, disconnecting, idle. ASSOS_STATE can be: NOT_ASSOCIATED/ASSOCIATING/ASSOCIATED.
CON_CHG=(status)	Displayed on change in connection state. Possible status values: LINK_LOSS - link loss DISCONN_LOCAL - locally initiated disconnect DISCONN_REMOTE - remotely initiated disconnect CONNECTED - connected to a Mesh control device
RCV=(string)	Displays raw data received over the mesh.
ASSOC_CHG=(state)	Response to change in association state. Possible state values: ASSOCIATING - association procedure started ASSOCIATED - association successfully complete UNASSOCIATED - unassociation complete

Table 7: BC Mesh prompts

Changelog

- BC Mesh 1.0
  - Initial release
  - Known limitations:
    - It is sometimes not possible to associate with a node, unless the phone is directly connected to it
    - The broadcasted messages are sometimes not reliably forwarded to the other nodes