UM1686
User manual

BlueNRG development kit

Introduction

This document describes how to use the BlueNRG development kit based on BlueNRG network processor. The kit is composed of a BlueNRG module (daughterboard) and an STM32L-based motherboard. The software package includes a graphical user interface application to control the BlueNRG through a simple ACI protocol.

Contents

1 Getting started

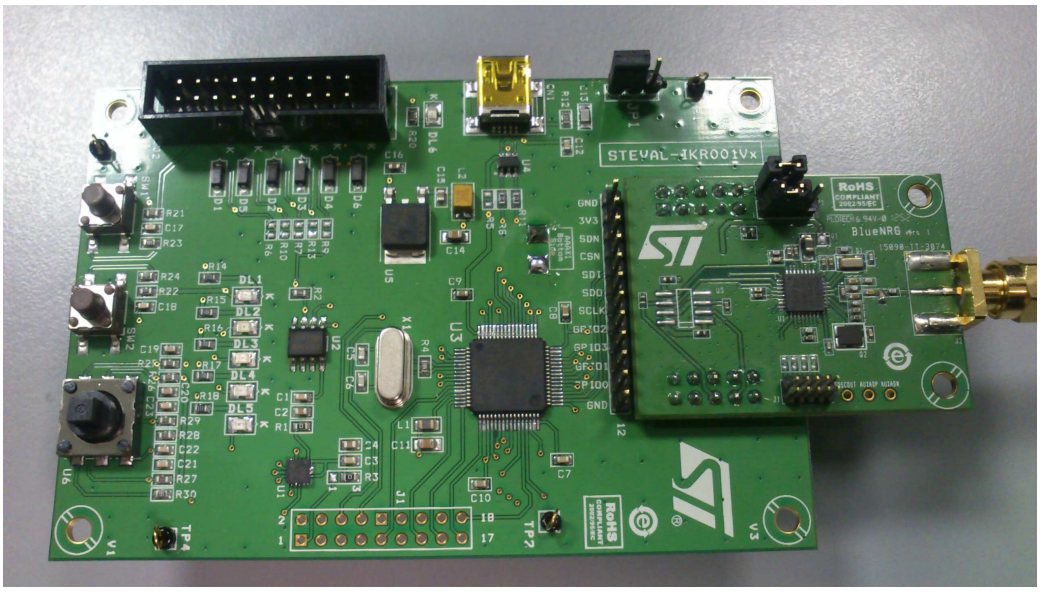
This section describes all the software and hardware requirements for running the BlueNRG GUI utility as well as the related installation procedure.

1.1 Kit contents

The kit is composed of the following items:

- 1 x development motherboard
- 1 x BlueNRG daughterboard
- 1 x 2.4 GHz Bluetooth antenna
- 1 x USB cable

Figure 1. BlueNRG kit motherboard with daughterboard connected



1.2 System requirements

The BlueNRG graphical user interface utility has the following minimum requirements:

- PC with Intel® or AMD® processor running one of the following Microsoft® operating systems:
  - Windows XP SP3
  - Windows Vista
  - Windows 7
- At least 128 Mb of RAM
- 2 x USB port
- 40 Mb of hard disk space available
- Adobe Acrobat Reader 6.0 or later.

1.3 BlueNRG development kit setup

- Extract the content of BlueNRG_DK_-x.x.x-Setup.zip file into a temporary directory.
- Launch the BlueNRG-DK-x.x.x-Setup.exe file and follow the on-screen instructions.

1.4 Hardware setup

1. Plug the BlueNRG daughterboard into the J4 and J5 connectors as shown in *Figure 1*.
2. Ensure the jumper configuration on the daughterboard is as in *Figure 1*.
3. Connect the motherboard to the PC with a USB cable (through connector CN1).
4. Verify that the PWR LED light is on.
5. Verify the new COM port is listed in the Port box in the left upper section of the GUI (click the combo box to update the list of system COM ports).

2 Hardware description

The development kit includes a generic motherboard and a BlueNRG daughterboard. The following sections describe these two components.

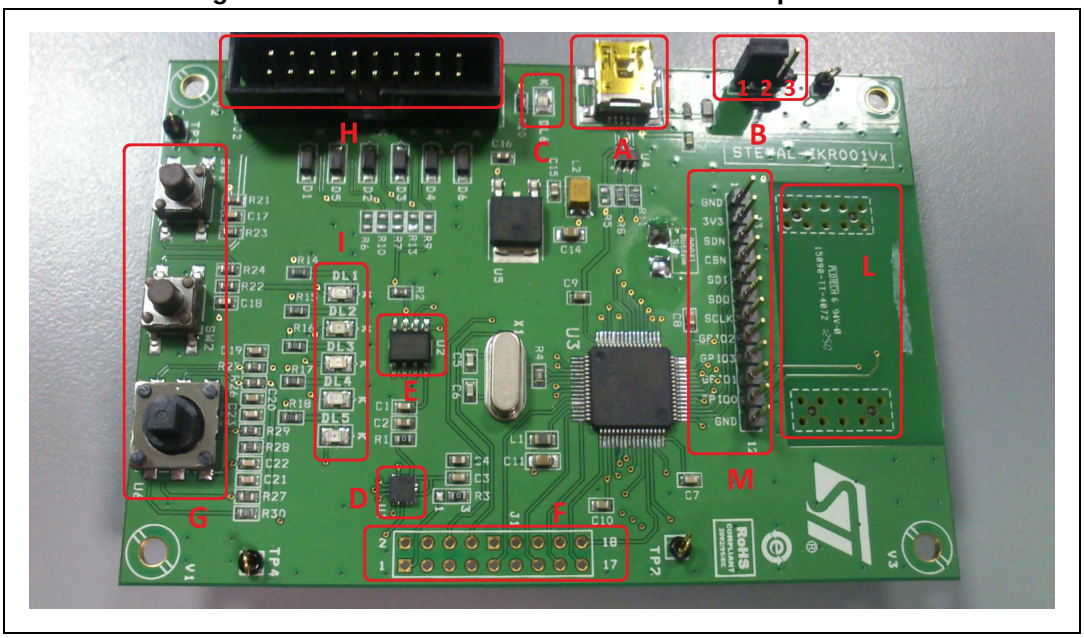
2.1 Motherboard

The motherboard included in the development kit allows testing of the functionality of the BlueNRG processor. The board can be used as a simple interface between the BlueNRG and a GUI application on the PC. The STM32L microcontroller on the board can also be programmed, so the board can be used to develop applications using the BlueNRG. A connector on the motherboard (*Figure 2*) allows access to the JTAG interface for programming and debugging. The board can be powered through a mini-USB connector that can also be used for I/O interaction with a USB Host. The board includes sensors, and buttons and a joystick for user interaction. The RF daughterboard can be easily connected through a dedicated interface.

This is the list of some of the features that are available on the boards:

- STM32L151RBT6 64-pin microcontroller
- Mini USB connector for power supply and I/O
- JTAG connector
- RF daughterboard interface
- One RESET button and one USER button
- One LIS3DH accelerometer
- One STLM75 temperature sensor
- One joystick
- 5 LEDs
- One PWR LED
- One battery holder for 2 AAA batteries
- One row of test points on the interface to the RF daughterboard

Figure 2. Motherboard for the BlueNRG development kit



2.1.1 Microcontroller and connections

The board features an STM32L151RB microcontroller, which is an ultra low-power microcontroller with 128 KB of Flash memory, 16 KB of RAM, 32-bit core ARM cortex-M3, 4 KB of data EEPROM, RTC, LCD, timers, USART, I^2C, SPI, ADC, DAC and comparators.

The microcontroller is connected to various components such as buttons, LEDs and connectors for external circuitry. The following table shows which functionality is available on each microcontroller pin.

Table 1. MCU pin description versus board function

Pin name	Pin	Board function							
		LEDs	DB connector	Buttons / Joystick	Accelerometer	Temperature sensor	USB	JTAG	Ext. conn
VLCD	1								
PC13	2		DB_SDN_RST						
PC14	3								3
PC15	4								5
OSC_IN	5								
OSC_OUT	6								
NRST	7			RESET					7
PC0	8	LED1							
PC1	9	LED2							
PC2	10		DB_PIN3						

Table 1. MCU pin description versus board function (continued)

Pin name	Pin num.	Board function							
		LEDs	DB connector	Buttons / Joystick	Accelerometer	Temperature sensor	USB	JTAG	Ext. conn
PC3	11								9
VSSA	12								
VDDA	13								
PA0	14								11
PA1	15								13
PA2	16								15
PA3	17								17
VSS_4	18								
VDD_4	19								
PA4	20				SPI1_NSS				
PA5	21				SPI1_SCK				
PA6	22				SPI1_MISO				
PA7	23				SPI1_MOSI				
PC4	24	LED4							
PC5	25	LED5							
PB0	26			JOY_DOWN					
PB1	27			JOY_RIGHT					
PB2	28								18
PB10	29				INT1				
PB11	30				INT2				
VSS_1	31								
VDD_1	32								
PB12	33		DB_CSN(1)						
PB13	34		DB_SCLK(1)						
PB14	35		DB_SDO(1)						
PB15	36		DB_SDI(1)						
PC6	37			PUSH_BTN					
PC7	38		DB_IO0(1)						
PC8	39		DB_IO1(1)						
PC9	40		DB_IO2(1)						
PA8	41			JOY_LEFT					
PA9	42			JOY_CENTER					

Table 1. MCU pin description versus board function (continued)

Pin name	Pin num.	Board function							
		LEDs	DB connector	Buttons / Joystick	Accelerometer	Temperature sensor	USB	JTAG	Ext. conn
PA10	43			JOY_UP					
PA11	44						USB_DM		
PA12	45						USB_DP		
PA13	46							JTMS	16
VSS_2	47								
VDD_2	48								
PA14	49							JTCK	14
PA15	50							JTDI	12
PC10	51		DB_IO3_IRQ(1)						
PC11	52		DB_PIN1						
PC12	53		DB_PIN2						
PD2	54	LED3							
PB3	55							JTDO	10
PB4	56							JNTRST	8
PB5	57					TSEN_INT			
PB6	58					I2C1_SCL			
PB7	59					I2C1_SDA			
BOOT0	60								
PB8	61								4
PB9	62								6
VSS_3	63								
VDD_3	64								

1. These lines are also available on the test point row

2.1.2 Power

The board can be powered either by the mini USB connector CN1 (A in *Figure 2*) or by 2 AAA batteries. To power the board through USB bus, jumper JP1 must be in position 1-2, as in *Figure 2* (B). To power the board using batteries, 2 AAA batteries must be inserted in the battery holder at the rear of the board, and jumper JP1 set to position 2-3.

When the board is powered, the green LED DL6 is on (C).

If needed, the board can be powered by an external DC power supply. Connect the positive output of the power supply to the central pin of JP1 (pin 2) and ground to one of the four test point connectors on the motherboard (TP1, TP2, TP3 and TP4).

2.1.3 Sensors

Two sensors are available on the motherboard:

- LIS3DH, an ultra-low power high performance three-axis linear accelerometer (D in *Figure 2*). The sensor is connected to the STM32L through the SPI interface. Two lines for interrupts are also connected.
- STLM75, a high precision digital CMOS temperature sensor, with I^2C interface (E in *Figure 2*). The pin for the alarm function is connected to one of the STM32L GPIOs.

2.1.4 Extension connector

There is the possibility to solder a connector on the motherboard to extend its functionality (F in *Figure 2*). 16 pins of the microcontroller are connected to this expansion slot (*Table 1*).

2.1.5 Push-buttons and joystick

For user interaction the board has two buttons. One is to reset the microcontroller, while the other is available to the application. There is also a digital joystick with 4 possible positions (left, right, up, down) (G in *Figure 2*).

2.1.6 JTAG connector

A JTAG connector on the board (H in *Figure 2*) allows the programming and debugging of the STM32L microcontroller on board[(a)], using an in-circuit debugger and programmer such as the ST-LINK/V2.

2.1.7 LEDs

Five LEDs are available (I in *Figure 2*).

- DL1: green
- DL2: orange
- DL3: red
- DL4: blue
- DL5: yellow

2.1.8 Daughterboard interface

The main feature of the motherboard is the capability to control an external board, connected to the J4 and J5 connectors (L in *Figure 2*). *Table 1* shows which pins of the microcontroller are connected to the daughterboard.

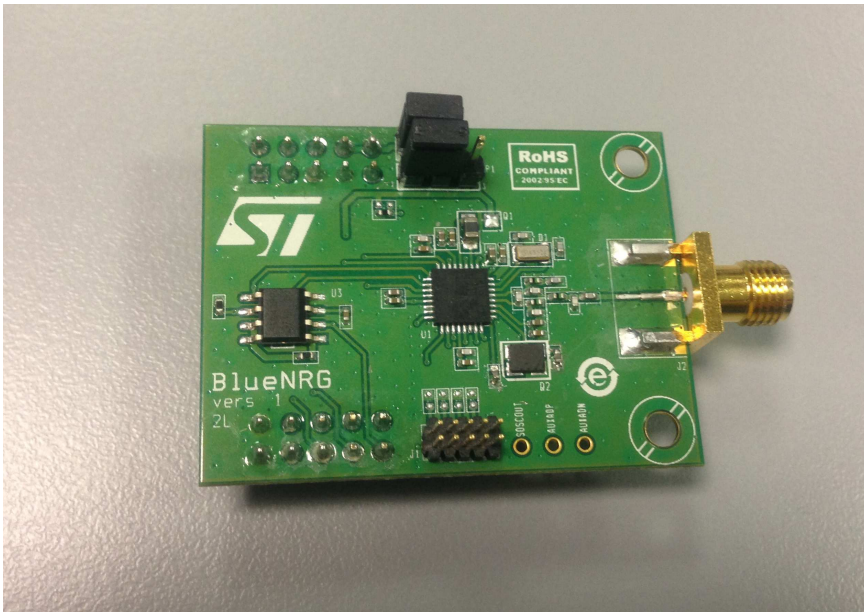
Some of the lines are connected also to a row of test points (M).

a. The STM32L is preprogrammed with DFU firmware that allows the downloading of a firmware image without the use of a programmer.

2.2 BlueNRG daughterboard

The BlueNRG daughterboard (*Figure 3*) included in the development kit is a small circuit board to be connected to the main board. It contains the BlueNRG network processor (in a QFN32 package), an SMA antenna connector, discrete passive components for RF matching and balun, and small number of additional components required by the BlueNRG for proper operation (see the schematic diagram in *Figure 10*).

Figure 3. BlueNRG daughterboard



The main features of the BlueNRG daughterboard are:

- BlueNRG low power network processor for Bluetooth low energy, with embedded host stack
- High frequency 16 MHz crystal
- Low frequency 32 kHz crystal for the lowest power consumption
- Balun, matching network and harmonic filter
- SMA connector

The daughterboard is also equipped with a discrete inductor for the integrated high-efficiency DC-DC converter, for best-in-class power consumption. It is still possible to disable the DC-DC converter. In this case the following changes must be performed on the daughterboard (see *Figure 10*):

- Remove inductor from solder pads 1 and 2 of D1
- Place a 0 ohm resistor between pads 1 and 3
- Move resistor on R2 to R1

For proper operation, jumpers must be set as indicated in *Figure 3*.

The following tables show the connections between the daughterboard and the main board.

Table 2. Connections between BlueNRG board and motherboard on left connector

Pin	J4 motherboard	J3 daughterboard
1	DB_PIN1	NC
2	3V3	3V3
3	DB_PIN3	NC
4	NC	NC
5	GND	GND
6	DB_PIN2	nS
7	GND	GND
8	3V3	U2 pin 1
9	DB_SDN_RST	RST
10	3V3	U2 pin 1

Table 3. Connections between BlueNRG board and motherboard on right connector

Pin	J5 motherboard	J4 daughterboard
1	GND	GND
2	GND	GND
3	DB_CSN	CSN
4	DB_IO3_IRQ	IRQ
5	DB_SCLK	CLK
6	DB_IO2	NC
7	DB_SDI	MOSI
8	DB_IO1	NC
9	DB_SDO	MISO
10	DB_IO0	NC

2.2.1 Current measurements

To monitor power consumption of the entire BlueNRG daughterboard, remove the jumper from U2 and insert an ammeter between pins 1 and 2 of the connector. Since power consumption of the BlueNRG during most operation time is very low, an accurate instrument in the range of few microamps may be required.

3 GUI software description

The BlueNRG GUI included in the software package is a graphical user interface that can be used to interact and evaluate the capabilities of the BlueNRG network processor.

This utility can send standard and vendor-specific HCI commands to the controller and receive events from it. It lets the user configure each field of the HCI command packets to be sent and analyzes all received packets. In this way BlueNRG can be easily managed at low level.

3.1 Requirements

In order to use the BlueNRG GUI, make sure you have correctly set up your hardware and software (BlueNRG GUI installed). The STM32L in the kit has been preprogrammed with a demo application (see *Section 5*). Hence, new firmware must be loaded into the STM32L. Firmware images can be found within the firmware folder. The firmware image that must be programmed is BlueNRG_VCOM.hex. The GUI has the ability to Flash new firmware.

In order to download binary images into the internal Flash of the STM32L, the microcontroller must be put into a special DFU (device firmware upgrade) mode. To enter DFU mode:

- Power up the board
- Press and hold USER button
- Reset the board using RESET button (keep USER button pressed while resetting) The orange LED DL2 will start to blink
- Release USER button
- Use BlueNRG GUI to Flash the device with new firmware (Tools -> Update Motherboard FW).

3.2 The BlueNRG graphical user interface

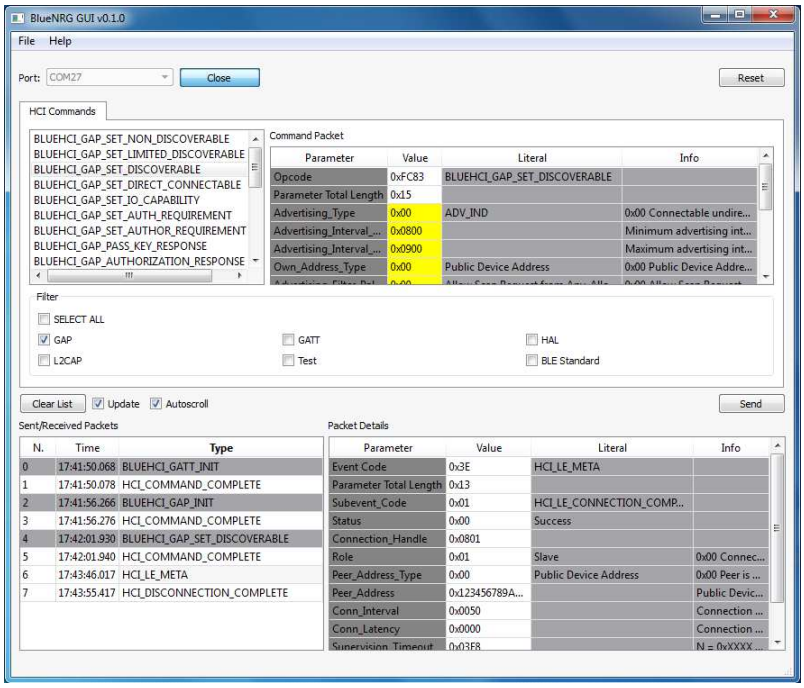
This section describes the main functions of BlueNRG GUI application.

You can run this utility by clicking on the BlueNRG GUI icon on the Desktop or under:

Start ⟶ STMicroelectronics ⟶ BlueNRG DK X.X.X ⟶ BlueNRG GUI

3.2.1 GUI main window

Figure 4. BlueNRG GUI main window



The BlueNRG GUI main window is characterized by different zones. Some of these zones can be resized.

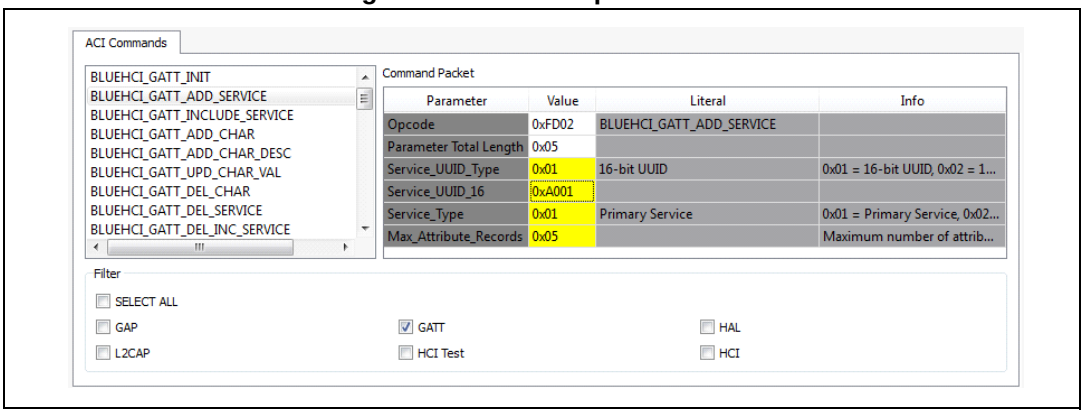
Port and interface selection

The uppermost zone allows the user to open the COM port associated to the BTLE controller.

HCI commands

The HCI Commands tab contains a list of all the available HCI commands. Commands can be filtered by checking/unchecking boxes under the filter section. After clicking on one of the commands, all the packet fields will be displayed on the command packet table in the upper-right section of the tab (see *Figure 5*).

Figure 5. Command packet table



The command packet table contains four columns:

- **Parameter**: name of the packet field as they are named in volume 2, part E of Bluetooth specification.
- **Value**: field value represented in hexadecimal format (right-click on a cell to change its representation format).
- **Literal**: meaning of the current field value.
- **Info**: description of the corresponding field.

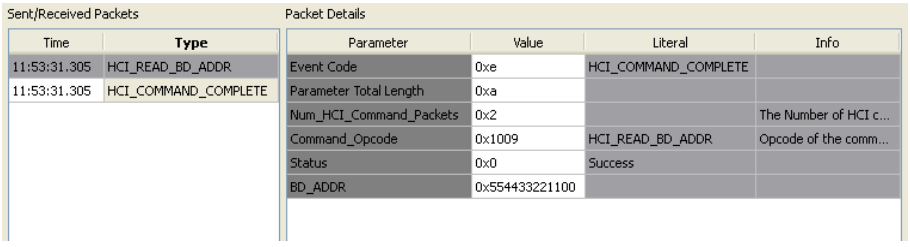
Only the yellow cells of this table can be modified by the user. The Parameter Total Length is fixed or automatically calculated after modifying cell content.

After the fields have been modified (if required) the command can be sent using the Send button.

HCI Packet history and details

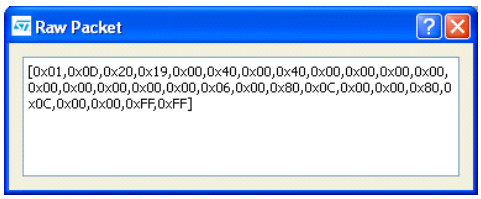
At the bottom of the main window, two tables show packets sent to and received from the BTLE controller, as well as other events. The left table (sent/received packets) holds a history of all packets (see *Figure 6*). The right one (packet details) shows all the details of the selected packet as is done in the command packet table (*Figure 6*).

Figure 6. Packet history and details



Double-clicking on a row of the sent/received packets table shows the raw packet.

Figure 7. Raw packet dump



Some events (displayed in yellow cells) can provide other information. HCI packets sent towards the BTLE controller are displayed in gray cells while received packets are shown inside white cells.

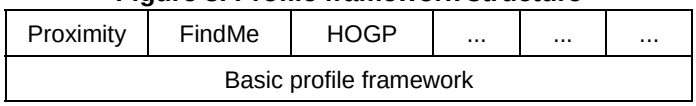
The Sent/received packets table can be cleared by clicking on clear list button. Update and auto-scrolling check boxes enable or disable updating and auto-scrolling of the Sent/received packets table while new packets are sent or received (however, information will still be printed).

4 Programming with BlueNRG network processor

The BlueNRG provides a high level interface to control its operation. This interface is called ACI (application-controller interface). The ACI is implemented as an extension to the standard Bluetooth HCI interface. Since BlueNRG is a network processor, the stack runs inside the device itself. Hence, no library is required on the external microcontroller, except for profiles and all the functions needed to communicate with the BlueNRG SPI interface.

The development kit software includes sample code that shows how to configure BlueNRG and send commands or parsing events. The source library is called Simple BlueNRG HCI to distinguish it from the library for the complete Profile Framework (not present in the software development kit). This library is able to handle multiple profiles at the same time and supports several Bluetooth GATT-based profiles for BlueNRG. Documentation on the ACI is provided in a separate document.

Figure 8. Profile framework structure



4.1 Requirements

In order to communicate with BlueNRG network processor very few resources are needed by the main processor. These are listed below:

- SPI interface
- Platform-dependent code to write/read to/from SPI
- A timer to handle SPI timeouts or to run Bluetooth LE Profiles

Minimum requirements in terms of Flash and RAM space largely depend on the functionality needed by the application, on the microprocessor that will run the code and on the compiler toolchain used to build the firmware.

On the STM32L (Cortex-M3 core), the memory footprint for the code interfacing the BlueNRG requires few kilobytes of Flash and RAM (typically 2-4 KB of Flash, and 0.8-1.5 KB of RAM). So a complete simple application (like the BlueNRG sensor demo) could require just 15 KB of Flash and 2 KB of RAM.

If using the complete BlueNRG Profile framework, the memory footprint is around 9 KB of code and 3 KB of data for just the ACI interface and the profile framework functions. The memory required for the profiles can vary depending on the complexity of the profile itself. For example, code for HID-over-GATT host is around 6 KB, while for heart rate monitor is around 2.3 KB.

4.2 Software directory structure

The project folder contains some sample code that can be used on the application processor to control the BlueNRG. Platform-dependent code is also provided for STM32L1 platforms. The example project provided in the package will run “as is” on the development kit.

The files are organized using the following folder structure:

- **Bluetooth LE.** Contains the code that is used to send ACI commands to the BlueNRG network processor. It contains also definitions of BlueNRG events.
- **platform.** Contains all the platform-dependent files. These can be taken as an example to build applications that can be run on other platforms.
- **examples.** Contains source code that can be used as an example to build other applications that will use the Bluetooth technology with the BlueNRG. Project files for IAR embedded workbench are also available.

5 BlueNRG sensor profile demo

The software development kit contains an example, which implements a proprietary Bluetooth profile: the sensor profile. This example is useful for building new profiles and applications that use the BlueNRG network processor. This GATT profile is not compliant to any existing specification. The purpose of this project is simply to show how to implement a given profile.

This profile exposes two services: acceleration service and environmental service. *Figure 9* shows the whole GATT database, including the GATT and GAP services that are automatically added by the stack.

One of the acceleration service's characteristics has been called free-fall characteristic. This characteristic cannot be read or written but can be notified. The application will send a notification on this characteristic (with value equal to 0x01) if a free-fall condition has been detected by the LIS3DH MEMS sensor (the condition is detected if the acceleration on the 3 axes is near zero for a certain amount of time). Notifications can be enabled or disabled by writing on the related client characteristic configuration descriptor.

The other characteristic exposed by the service gives the current value of the acceleration that is measured by the accelerometer. The value is made up of six bytes. Each couple of bytes contains the acceleration on one of the 3 axes. The values are given in mg. This characteristic is readable and can be notified if notifications are enabled.

Another service is also defined. This service contains characteristics that expose data from some environmental sensors: temperature, pressure and humidity[(b)]. For each characteristic, a characteristic format descriptor is present to describe the type of data contained inside the characteristic. All of the characteristics have read-only properties

b. An expansion board with LPS25H pressure sensor and HTS221 humidity sensor can be connected to the motherboard through the expansion connector (F in *Figure 2*). If the expansion board is not detected, only temperature from STLM75 will be used.

Figure 9. Demo GATT

#	Handle	UUID (16 or 128bit)	Attribute Type	Properties								Initial Parameter Value
				BRD	RD	WR NO RESP	WR	NOT	IND	SIGN WR	EXT	
1	0001	2800	Primary Service									{Service=0x1801 ("Attribute Profile")}
2	0002	2803	Characteristic						X			{handle=0x0003, UUID=0x2A05}
3	0003	2A05	**Service Changed**									{start handle=0x0001, end handle=0xFFFF}
4	0004	2902	Client Characteristic Configuration									0x0000
5	0005	2800	Primary Service									{Service=0x1800 ("Generic Access Profile")}
6	0006	2803	Characteristic		X	X	X			X		{handle=0x0007, UUID=0x2A00}
7	0007	2A00	**Device Name**									"bluenrg"
8	0008	2803	Characteristic		X	X	X					{handle=0x0009, UUID=0x2A01}
9	0009	2A01	**Appearance**									0x0000
16	0010	2800	Primary Service									{Service=0x02366E80CF3A11E19AB40002A5D5C51B ("Acc Service")}
17	0011	2803	Characteristic					X				{handle=0x0012, UUID=0xE23E78A0CF4A11E18FFC0002A5D5C51B}
18	0012	E23E78A0CF4A11E18FFC0002A5D5C51B	**Free Fall**									0x00
19	0013	2902	Client Characteristic Configuration									0x0000
20	0014	2803	Characteristic		X			X				{handle=0x0015, UUID=0x340A1B80CF4B11E1AC360002A5D5C51B}
21	0015	340A1B80CF4B11E1AC360002A5D5C51B	**Acceleration**									0x000000000000
22	0016	2902	Client Characteristic Configuration									0x0000
23	0017	2800	Primary Service									{Service=0x02366E80CF3A11E19AB40002A5D5C51B ("Env Service")}
24	0018	2803	Characteristic		X							{handle=0x0019, UUID=0xA32E5520E47711E2A9E30002A5D5C51B}
25	0019	A32E5520E47711E2A9E30002A5D5C51B	**Temperature**									0x0000
26	0075	2904	Characteristic Format									{format=0x0E, exp=-1, unit=0x272F, n_sp=0x00, descr=0x0000}
27	001B	2803	Characteristic		X							{handle=0x001C, UUID=0xCD20C480E48B11E2840B0002A5D5C51B}
28	001C	CD20C480E48B11E2840B0002A5D5C51B	**Pressure**									0x000000
29	0075	2904	Characteristic Format									{format=0x0F, exp=-5, unit=0x2780, n_sp=0x00, descr=0x0000}
30	001E	2803	Characteristic		X							{handle=0x001F, UUID=0x01C50B60E48C11E2A0730002A5D5C51B}
31	001F	01C50B60E48C11E2A0730002A5D5C51B	**Humidity**									0x0000
32	0075	2904	Characteristic Format									{format=0x06, exp=-1, unit=0x2700, n_sp=0x00, descr=0x0000}

5.1 BlueNRG app for smartphones

An application is available for smartphones (iOS and android), that works with the sensor profile demo. The development kits are preprogrammed with the sensor profile demo firmware. If the development board has been flashed with another firmware, it can be programmed with the correct firmware. Refer to *Section 4.1* for the programming procedure using the device firmware upgrade feature and BlueNRG GUI. The correct pre-compiled firmware can be found inside firmware folder (SensorDemo.hex). The source file for the demo is inside the project folder.

This app enables notifications on the acceleration characteristic and displays the value on the screen. Data from environmental sensors are also periodically read and displayed.

Figure 10. BlueNRG app

6 Connection with a central device

This section describes how to interact with a central device, while BlueNRG is acting as a peripheral. The central device can be another BlueNRG acting as a master, or any other Bluetooth smart or smart-ready device.

First, BlueNRG must be set up. In order to do this, a series of ACI command need to be sent to the processor.

6.1 Initialization

BlueNRG's stack must be correctly initialized before establishing a connection with another Bluetooth LE device. This is done with two commands:

- BLUEHCI_GATT_INIT
- BLUEHCI_GAP_INIT(Role=0x01)

See ACI documentation for more information on these commands and on those that follow as well. Peripheral role must be specified inside the GAP_INIT command.

6.2 Add service and characteristics

BlueNRG's Bluetooth LE stack has both server and client capabilities. A characteristic is an element in the server database where data are exposed. A service contains one or more characteristics. Add a service using the following command. Parameters are provided only as an example.

- BLUEHCI_GATT_ADD_SERVICE(Service_UUID_Type=0x01, Service_UUID_16=0xA001, Service_Type=0x01, Max_Attributes_Records=0x06)

The command will return the service handle (e.g., 0x0010). A characteristic must now be added to this service. This service is identified by the service handle.

- BLUEHCI_GATT_ADD_CHAR(Service_Handle=0x0010, Char_UUID_Type=0x01, Char_UUID_16=0xA002, Char_Value_Length=10, Char_Properties=0x1A, Security_Permissions=0x00, GATT_Evt_Mask=0x01, Enc_Key_Size=0x07, Is_Variable=0x01)

With this command a variable-length characteristic has been added, with read, write and notify properties. The characteristic handle is also returned (Char_Handle).

6.3 Set security requirements

BlueNRG exposes a command that the application can use to specify its security requirements. If a characteristic has security restrictions, a pairing procedure must be initiated by the central in order to access that characteristic. Let's assume we want the user to insert a passcode during the pairing procedure.

- BLUEHCI_GAP_SET_AUTH_REQUIREMENT(MITM_Mode=0x01, OOB_Enable=0, OOB_Data=0, Min_Encryption_Key_Size=7, Max_Encryption_Key_Size=16, Use_Fixed_Pin=0, Fixed_Pin=123456, Bonding_Mode=1)

6.4 Enter connectable mode

Use GAP ACI commands to enter one of the discoverable and connectable modes.

– BLUEHCI_GAP_SET_DISCOVERABLE(Advertising_Type=0x00, Advertising_Interval_Min=0x800, Advertising_Interval_Max=0x900, Own_Address_Type=0x00, Advertising_Filter_Policy=0x00, Local_Name_Length=0x08, Local_Name='\x08BlueNRG', Service_UUID_Length=0x00, Service_UUID_List=0x00, Slave_Connection_Interval_Min=0x0000, Slave_Connection_Interval_Max=0x0000)

The Local_Name parameter contains the name that will be present in advertising data, as described in Bluetooth core specification version 4.0, Vol. 3, Part C, Ch. 11[(c)].

6.5 Connection with central device

Once BlueNRG is put in a discoverable mode, it can be seen by a central device in scanning.

Any Bluetooth smart and smart-ready device can connect to BlueNRG, such as a smartphone. LightBlue is one of the applications in the Apple Store for iPhone® 4S/5 and later versions of Apple's iPhone.

Start the LightBlue application. It will start to scan for peripherals. A device with the BlueNRG name will appear on the screen. Tap on the box to connect to the device. A list of all the available services will be shown on the screen. Touching a service will show the characteristics for that service.

BlueNRG has added two standard services: GATT Service (0x1801) and GAP service (0x1800).

Try to read the characteristic from the service we have just added (0xA001). The characteristic has a variable length attribute, so you will not see any value. Write a string into the characteristic and read it back.

BlueNRG can send notifications of the characteristic that has been previously added, with UUID 0xA002 (after notifications have been enabled). This can be done using the following command:

– BLUEHCI_GATT_UPD_CHAR_VALUE(Service_Handle=0x0010, Char_Handle=0x0011, Val_Offset=0, Char_Value_Length=0x05, Char_Value='hello')

Once this ACI command has been sent, the new value of the characteristic will be displayed on the phone.

c. The first byte of the value is the AD Type. In BlueNRG GUI the \xHH notation is used to specify a byte in hexadecimal format inside a string.

7 Revision history

Table 4. Document revision history

Date	Revision	Changes
28-Nov-2013	1	Initial release.

Please Read Carefully:

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST’s terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

UNLESS OTHERWISE SET FORTH IN ST’S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

ST PRODUCTS ARE NOT DESIGNED OR AUTHORIZED FOR USE IN: (A) SAFETY CRITICAL APPLICATIONS SUCH AS LIFE SUPPORTING, ACTIVE IMPLANTED DEVICES OR SYSTEMS WITH PRODUCT FUNCTIONAL SAFETY REQUIREMENTS; (B) AERONAUTIC APPLICATIONS; (C) AUTOMOTIVE APPLICATIONS OR ENVIRONMENTS, AND/OR (D) AEROSPACE APPLICATIONS OR ENVIRONMENTS. WHERE ST PRODUCTS ARE NOT DESIGNED FOR SUCH USE, THE PURCHASER SHALL USE PRODUCTS AT PURCHASER’S SOLE RISK, EVEN IF ST HAS BEEN INFORMED IN WRITING OF SUCH USAGE, UNLESS A PRODUCT IS EXPRESSLY DESIGNATED BY ST AS BEING INTENDED FOR “AUTOMOTIVE, AUTOMOTIVE SAFETY OR MEDICAL” INDUSTRY DOMAINS ACCORDING TO ST PRODUCT DESIGN SPECIFICATIONS. PRODUCTS FORMALLY ESCC, QML OR JAN QUALIFIED ARE DEEMED SUITABLE FOR USE IN AEROSPACE BY THE CORRESPONDING GOVERNMENTAL AGENCY.

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

ST and the ST logo are trademarks or registered trademarks of ST in various countries.
Information in this document supersedes and replaces all information previously supplied.
The ST logo is a registered trademark of STMicroelectronics. All other names are the property of their respective owners.

STMicroelectronics group of companies

Australia - Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Philippines - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

www.st.com