BlueNRG-1, BlueNRG-2 development kits

Introduction

The BlueNRG-1, BlueNRG-2 devices are low power Bluetooth® smart system on chip, compliant with the Bluetooth® specification and supporting master, slave and simultaneous master-and-slave roles. Further, BlueNRG-2 supports the Bluetooth Low Energy extended packet length feature.

The following BlueNRG-1, BlueNRG-2 kits are available:

1. BlueNRG-1 development platforms (order code: STEVAL-IDB007V1, STEVAL-IDB007V2)
2. BlueNRG-2 development platforms (order code: STEVAL-IDB008V1, STEVAL-IDB008V2)

The STEVAL-IDB007Vx, STEVAL-IDB008Vx also provide a set of hardware resources for a wide range of application scenarios: sensor data (accelerometer, pressure and temperature sensor), remote control (buttons and LEDs) and debug message management through USB virtual COM. Three power options are available (USB only, battery only and external power supply plus USB) for high application development and testing flexibility.

Figure 1. STEVAL-IDB007V1 development platform

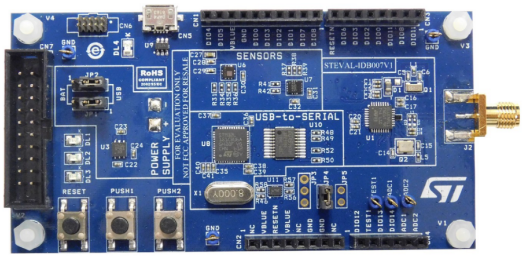


Figure 2. STEVAL-IDB007V2 development platform

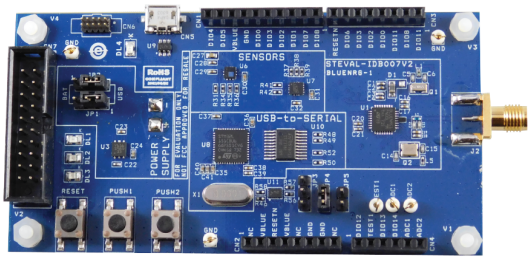


Figure 3. STEVAL-IDB008V1 development platform

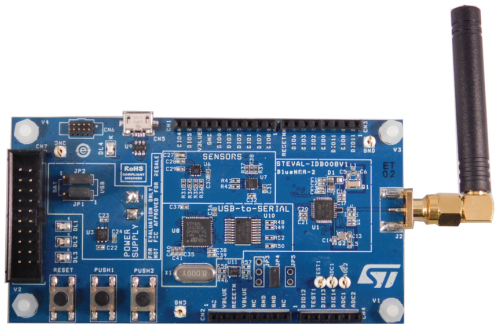
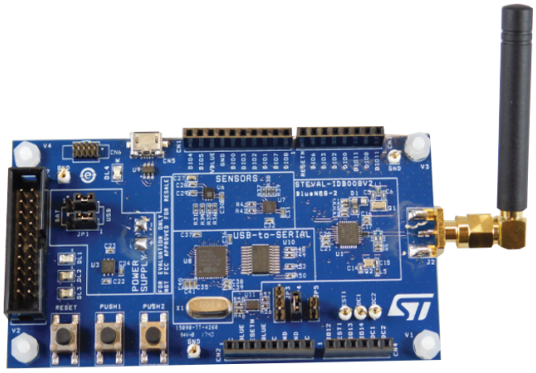


Figure 4. STEVAL-IDB008V2 development platform



1 Getting started

1.1 Kit contents

The STEVAL-IDB007Vx/STEVAL-IDB008Vx kits include respectively:

- 1 BlueNRG-1/BlueNRG-2 development platform
- 1 2.4 GHz Bluetooth antenna
- 1 USB cable

1.2 System requirements

The BlueNRG-1, BlueNRG-2 Navigator, Flasher and Radio Parameters Wizard PC applications require:

- 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
- USB ports
- At least 40 MB of available hard disk space
- Adobe Acrobat Reader 6.0 or later.

1.3 BlueNRG-1_2 development kit setup

The following BlueNRG-1, BlueNRG-2 DK software packages are available:

1. BlueNRG-1_2 DK SW package for BlueNRG-1, BlueNRG-2 BLE stack v2.x family (STSW-BLUENRG1-DK)

2. BlueNRG-1_V1 DK SW package for BlueNRG-1 BLE stack v1.x family (STSW-BNRG_V1-DK)

After downloading the selected software package (STSW-BLUENRG1-DK or STSW-BNRG_V1-DK) from www.st.com, extract BlueNRG-1_2_DK-x.x.x-Setup.zip or BlueNRG-1_V1_DK-x.x.x-Setup.zip contents to a temporary directory, launch BlueNRG-1_2-DK-x.x.x-Setup.exe or BlueNRG-1_V1-DK-x.x.x-Setup.exe and follow the on-screen instructions.

Note: EWARM Compiler 7.70 or later is required for building the BlueNRG1_2_DK_x.x.x, BlueNRG1_V1_DK_x.x.x demonstration applications.

Note: Keil MDK-ARM and Atollic-True Studio toolchains are also supported.

2 Hardware description

2.1 STEVAL-IDB007Vx/STEVAL-IDB008Vx boards overview

The BlueNRG-1/BlueNRG-2 devices in the STEVAL-IDB007Vx/STEVAL-IDB008Vx development kits lets you experiment with BlueNRG-1/BlueNRG-2 system on chip functions. They feature:

- Bluetooth® SMART board based on the BlueNRG-1/BlueNRG-2 Bluetooth low energy system on chip
- Associated development kit SW package including firmware and documentation
- Up to +8 dBm available output power (at antenna connector)
- Excellent receiver sensitivity (-88 dBm)
- Very low power consumption: 7.7 mA RX and 8.3 mA TX at -2 dBm
- Bluetooth® low energy compliant, supports master, slave and simultaneous master-and-slave roles
- Integrated balun which integrates a matching network and harmonics filter
- SMA connector for antenna or measuring equipment
- 3 user LEDs
- 2 user buttons
- 3D digital accelerometer and 3D digital gyroscope
- MEMS pressure sensor with embedded temperature sensor
- Battery holder
- JTAG debug connector
- USB to serial bridge for providing I/O channel with the BlueNRG-1/BlueNRG-2 device
- Jumper for measuring current for BlueNRG-1/BlueNRG-2 only
- RoHS compliant

The following figure and table describe physical sections of the board.

Figure 5. STEVAL-IDB007Vx board components

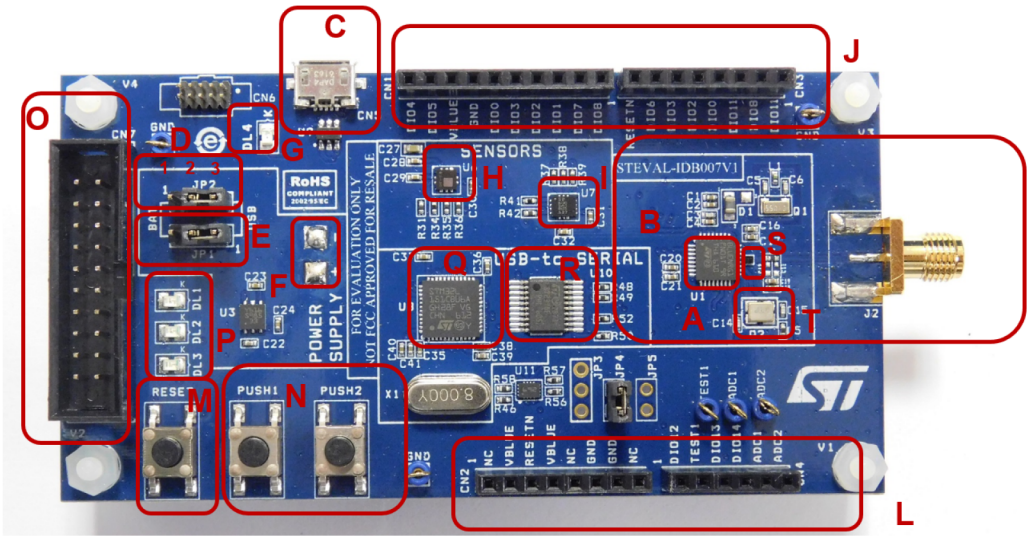


Figure 6. **STEVAL-IDB008Vx board components**

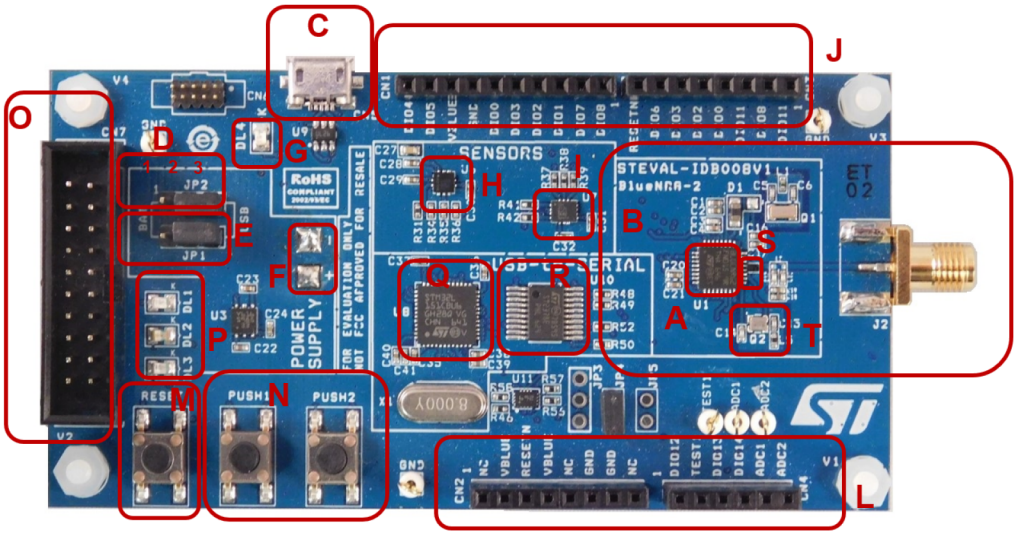


Table 1. **STEVAL-IDB007Vx/STEVAL-IDB008Vx board component descriptions**

Region	Description
A	BlueNRG-1 SoC on STEVAL-IDB007Vx BlueNRG-2 SoC on STEVAL-IDB008Vx
C	Micro USB connector for power supply and I/O
O	JTAG connector
M	RESET button
N	two USER buttons
H	LPS25HB MEMS pressure sensor with embedded temperature
I	LSM6DS3 3D digital accelerometer and 3D digital gyroscope
G	PWR LED
P	three user LEDs
back of the PCB	battery holder for two AAA batteries
J, L	Two rows of Arduino-compliant connectors
S	Integrated balun with matching network and harmonics filter (BALF-NRG-01D3 on STEVAL-IDB007V1/ STEVAL-IDB008V1 and BALF-NRG-02D3 on STEVAL-IDB007V2/STEVAL-IDB008V2)
Q	STM32L151CBU6 48-pin microcontroller (USB to serial bridge for I/O channel to PC communication) (1)
R	ST2378E level translator to adapt voltage level between STM32 and BlueNRG-1
T	16 MHz High Speed Crystal on STEVAL-IDB007Vx 32 MHz High Speed Crystal on STEVAL-IDB008Vx

1. *STM32 is not intended to be programmed by users*

2.2 BlueNRG-1, BlueNRG-2 SoC connections

The BlueNRG-1, BlueNRG-2 very low power Bluetooth low energy (BLE) single-mode system on chip (Figure 5. STEVAL-IDB007Vx board components – region A / Figure 6. STEVAL-IDB008Vx board components - region A)

have respectively 160 KB, 256 KB of Flash, 24 KB of RAM, a 32-bit core ARM cortex-M0 processor and several peripherals (ADC, GPIOs, I²C, SPI, Timers, UART, WDG and RTC).

The microcontroller is connected to various components such as buttons, LEDs and sensors. The following table describes the microcontroller pin functions.

Table 2. BlueNRG-1, BlueNRG-2 pins description with board functions

Pin name	Pin No	Board function									
		LED	Micro	Button	Pressure sensor	3D accelerometer and gyroscope	JTAG	Arduino connectors			
								CN1	CN2	CN3	CN4
DIO10	1						JTMS-SWTDIO				
DIO9	2						JTCK-SWTCK				
DIO8	3		TXD (PA2)					pin 1 IO8		pin 2 TX	
DIO7	4	DL2						pin 2 IO9			pin 6 SCL
DIO6	5	DL1								pin 7 IO6	pin 5 SDA
VBAT3	6										
DIO5	7				SDA PUSH2 button			pin 9 SDA			
DIO4	8				SCL			pin 10 SCL			
DIO3	9					SDO/SA0		pin 5 MISO		pin 6 IO5	
DIO2	10					SDA		pin 4 MOSI		pin 5 IO4	
DIO1	11					CS	JTAG-TDO	pin 3 CS			
DIO0	12					SCL	JTAG-TDI	pin 6 SCK		pin 4 IO3	
ANATEST0/DIO14	13	DL3									pin 4 AD3
ANATEST1	14										
ANATEST2	15										
ANATEST3	16										
FXTAL1	17										
FXTAL0	18										
VBAT2	19										
RF1	20										
RF0	21										
SXTAL1	22										
SXTAL0	23										
VBAT1	24										

Pin name	Pin No	Board function									
		LED	Micro	Button	Pressure sensor	3D accelerometer and gyroscope	JTAG	Arduino connectors			
								CN1	CN2	CN3	CN4
RESET	25		RESET	RESET			RESET		pin 3 NRST	pin 8 IO7	
SMPSFILT1	26										
SMPSFILT2	27										
VDD1V2	28										
DIO13	29			PUSH1							pin 3 AD2
DIO12	30					INT1					pin 1 AD0
FTEST	31										
DIO11	32		RXD PA3							pin 1 RX pin 3 IO2	pin 2 AD1

The board section labeled respectively BlueNRG-1, BlueNRG-2 (Figure 5. STEVAL-IDB007Vx board components, Figure 6. STEVAL-IDB008Vx board components – region B) includes the following main components:

- BlueNRG-1/BlueNRG-2 low power system on chip (in a QFN32 package)
- High frequency 16 MHz crystal on STEVAL-IDB007Vx and 32 MHz crystal on STEVAL-IDB008Vx
- Low frequency 32 kHz crystal for the lowest power consumption
- Integrated balun which integrates a matching network and harmonics filter
- SMA connector

For more details, see Figure 27. STEVAL-IDB007V1 arduino connectors and Figure 36. STEVAL-IDB008V1 circuit schematic arduino connectors .

2.3 Power supply

Green LED DL4 (Figure 5. STEVAL-IDB007Vx board components, Figure 6. STEVAL-IDB008Vx board components – region G) signals the board is being powered, either via:

- micro USB connector CN5 (Figure 5. STEVAL-IDB007Vx board components, Figure 6. STEVAL-IDB008Vx board components – region C)
- two AAA batteries (region F)
- an external DC power supply plus micro USB connector

The following table describes the power supply modes available on the STEVAL-IDB007V1, STEVAL-IDB008V1 boards and corresponding jumper settings.

Table 3. STEVAL-IDB007V1,STEVAL-IDB008V1 kits platforms power supply modes

Power supply mode	JP1	JP2	Comment
1 - USB	Fitted: 1-2	Fitted: 2-3	USB supply through connector CN5 (Figure 5. STEVAL-IDB007Vx board components, Figure 6. STEVAL-IDB008Vx board components – region C)
2 - Battery	Fitted: 2-3	Fitted: 1-2	The supply voltage must be provided through battery pins (region F).

Power supply mode	JP1	JP2	Comment
3 - Combo	Fitted: 1-2	Optional	USB supply through connector CN5 for STM32L1; JP2 pin 2 external power for BlueNRG-1, BlueNRG-2

2.4 Jumpers

The following jumpers are available:

Table 4. STEVAL-IDB007Vx,STEVAL-IDB008Vx kits platforms jumpers

Jumper	Description
JP1	1-2: to provide power from USB (JP2: 2-3) 2-3: to provide power from battery holder (JP2: 1-2)
JP2	1-2: to provide power from battery holder (JP1: 2-3) 2-3: to provide power from USB (JP1: 1-2) JP2 pin 2 to VDD to provide external power supply to BlueNRG-1, BlueNRG-2 (JP1: 1-2)
JP3	pin 1 and 2 UART RX and TX of MCU pin 3 GND
JP4	Fitted: to provide VBLUE to BlueNRG-1, BlueNRG-2. It can be used also for current measurement.
JP5	Fitted: TEST pin to VBLUE Not fitted: TEST pin to GND

2.5 Sensors

The following sensors are available on the platform:

1. An LPS25HB (Figure 5. STEVAL-IDB007Vx board components, Figure 6. STEVAL-IDB008Vx board components – region H) is a piezoresistive absolute pressure sensor which functions as a digital output barometer. The device comprises a sensing element and an IC interface which communicates through I²C from the sensing element to the application.
2. An LSM6DS3 3D (region I) digital accelerometer and 3D digital gyroscope with embedded temperature sensor which communicates via SPI interface. One line for interrupt is also connected.

2.6 Extension connector

BlueNRG-1, BlueNRG-2 signal test points are shared on two Arduino-compliant connector rows: CN1, CN3 (Figure 5. STEVAL-IDB007Vx board components, Figure 6. STEVAL-IDB008Vx board components – region J) and CN2, CN4 (region L). See Table 2. BlueNRG-1, BlueNRG-2 pins description with board functions.

2.7 Push-buttons

The board has one user button to reset the microcontroller (Figure 5. STEVAL-IDB007Vx board components, Figure 6. STEVAL-IDB008Vx board components – region M) and two further buttons for application purposes (region N).

2.8 JTAG connector

A JTAG connector (Figure 5. STEVAL-IDB007Vx board components, Figure 6. STEVAL-IDB008Vx board components – region O) allows BlueNRG-1, BlueNRG-2 microcontroller programming and debugging with an in-circuit debugger and programmer such as ST-LINK/V2.

Note: Only SWD mode is supported

2.9 LEDs

LEDs DL1 (yellow), DL2 (red), DL3 (blue) and DL4 (green, power LED) are available on the board (Figure 5. STEVAL-IDB007Vx board components, Figure 6. STEVAL-IDB008Vx board components – regions G and P).

2.10 STM32L151CBU6 microcontroller

The most important feature of the STM32L151CBU6 48-pin microcontroller (Figure 5. STEVAL-IDB007Vx board components, Figure 6. STEVAL-IDB008Vx board components – regions Q) is the USB to serial bridge providing an I/O channel with the BlueNRG-1, BlueNRG-2 device.

The microcontroller is connected to the BlueNRG-1, BlueNRG-2 device through an ST2378E level translator (region R).

Note: The STM32L microcontroller on the board is not intended to be programmed by users. ST provides a pre-programmed firmware image for the sole purpose of interfacing BlueNRG-1, BlueNRG-2 to a USB host device (e.g., a PC).

2.11 Current measurements

To monitor the power consumption of the BlueNRG-1, BlueNRG-2 only, remove the jumper from JP4 and insert an ammeter between pins 1 and 2 of the connector (when the power is ON, remove the USB connection).

Since power consumption of the BlueNRG-1, BlueNRG-2 are usually very low, an accurate instrument in the range of few micro amps is recommended.

2.12 Hardware setup

1. Connect an antenna to the SMA connector
2. Configure the board to USB power supply mode as per Table 3. STEVAL-IDB007V1,STEVAL-IDB008V1 kits platforms power supply modes
3. Connect the board to a PC via USB cable (connector CN5)
4. Verify the power indication LED DL4 is on.

3 BlueNRG-1, BlueNRG-2 Navigator

BlueNRG-1, BlueNRG-2 Navigator are user friendly GUI which lets you select and run demonstration applications easily, without requiring any extra hardware. With it, you can access the following DK software package components:

- BlueNRG-1, BlueNRG-2 Bluetooth low energy (BLE) demonstration applications
- BlueNRG-1, BlueNRG-2 peripheral driver examples
- BlueNRG-1, BlueNRG-2 development kits
- release notes
- license files

With BlueNRG-1 DK Navigator, you can directly download and run the selected prebuilt application binary image (BLE examples or peripheral driver example) on the BlueNRG-1, BlueNRG-2 platform without a JTAG interface.

The interface gives demo descriptions and access to board configurations and source code if needed.

User can run the utility through the BlueNRG-1 and BlueNRG-2 Navigator icon under:

Start → STMicroelectronics → BlueNRG -1_2 DK X.X.X → BlueNRG-1 Navigator, BlueNRG-2 Navigator or Start → STMicroelectronics → BlueNRG -1_V1 DK X.X.X → BlueNRG-1 Navigator.

Figure 7. BlueNRG-1 Navigator

Note: BlueNRG-1 Navigator and BlueNRG-2 Navigator are two instances of same application tailored for the specific selected device, in order to select the related available resources. Next sections focus on BlueNRG-1 Navigator, but same concepts are also valid for BlueNRG-2 Navigator.

3.1 BlueNRG-1 Navigator 'Demonstration Applications'

You can navigate the menus for the reference/demo application you want to launch. For each application, the following information is provided:

- Application settings (if applicable)
- Application description
- Application hardware related information (e.g., LED signals, jumper configurations, etc.)

The following functions are also available for each application:

- **Flash**: to automatically download and run the available prebuilt binary file to a BlueNRG-1 platform connected to a PC USB port.
- **Doc**: to display application documentation (html format)
- **Project**: to open the project folder with application headers, source and project files.

The figure below shows you how to run the BLE Beacon demo application; the other demos function similarly.

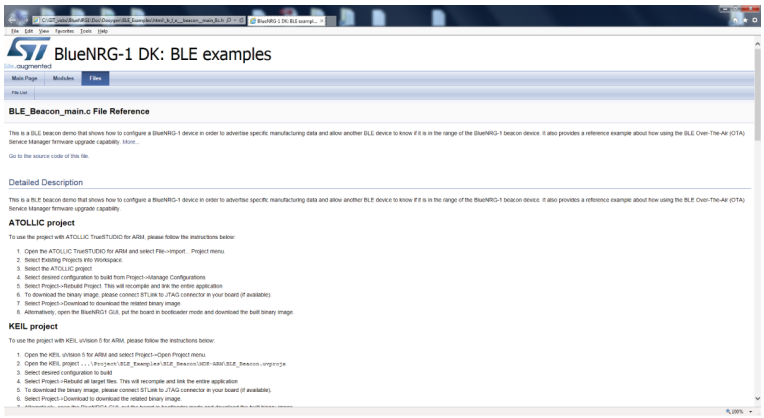
Figure 8. BLE Beacon application

When a BlueNRG-1 platform is connected to your PC USB port, you can press the "Flash & Run" tab on the selected application window to download and run the available prebuilt application binary image on the BlueNRG-1 platform.

Figure 9. BLE Beacon Flash programming

Selecting the “Doc” tab opens the relative html documentation.

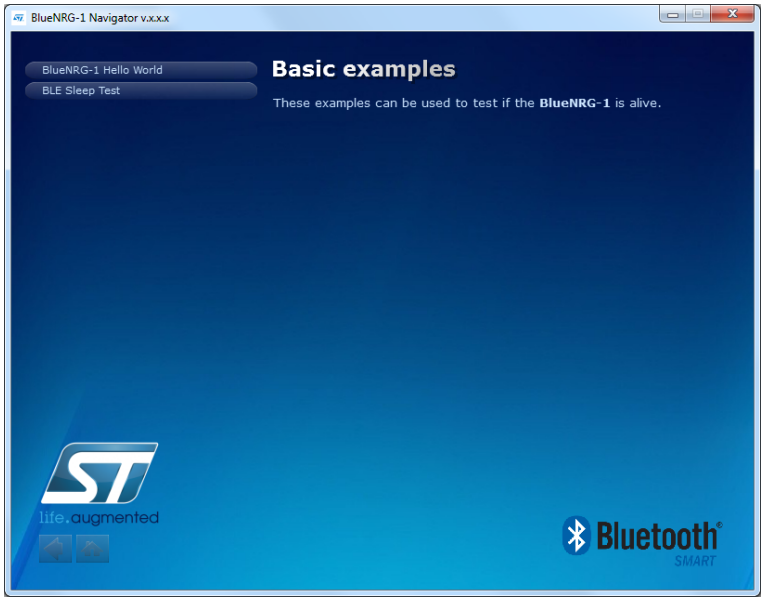
Figure 10. BLE Beacon documentation



3.1.1 BlueNRG-1 Navigator ‘Basic examples’

This page lists some basic sample applications for the BlueNRG-1 device to verify that BlueNRG-1 device is alive as well as the device sleep and wakeup modes.

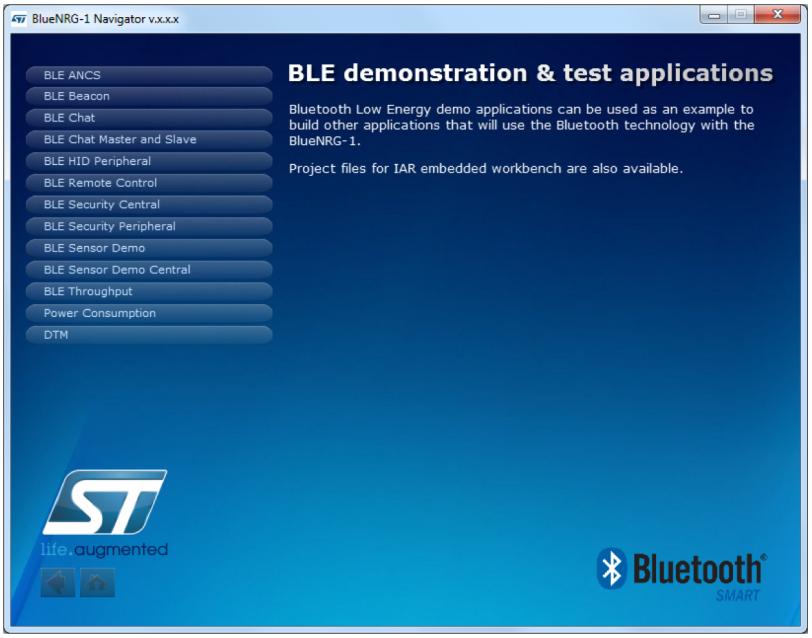
Figure 11. Basic examples



3.1.2 BlueNRG-1 Navigator ‘BLE demonstration and test applications’

This page lists all the available Bluetooth low energy (BLE) demonstration applications in the DK software package. These applications provide usage examples of the BLE stack features for the BlueNRG-1 device.

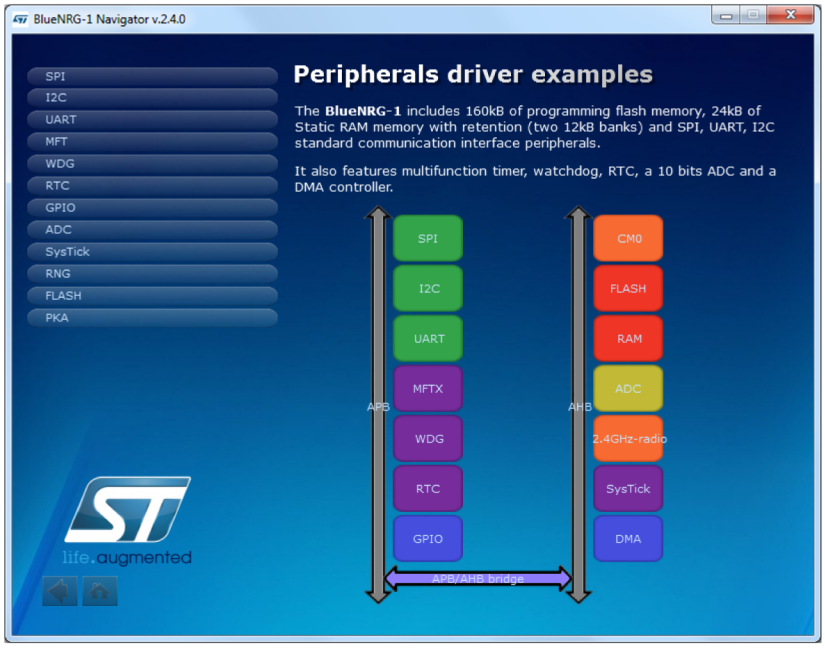
Figure 12. **BLE demonstration and test applications**



3.1.3 BlueNRG-1 Navigator ‘Peripherals driver examples’

This page lists the available BlueNRG-1 peripherals and corresponding test applications to work with certain features specific to the selected BlueNRG-1 peripheral.

Figure 13. **Peripherals driver examples**



3.2 BlueNRG-1 Navigator ‘Development Kits’

This window displays the available BlueNRG-1 DK Kit platforms and corresponding resources. When you hovers the mouse pointer over a specific item, the related component is highlighted on the board.

Figure 14. STEVAL-IDB007V1 Kit components

3.2.1 BlueNRG-1 Navigator ‘Release Notes’ and ‘License’

As their name suggests, these pages display the DK SW package Release Notes (html format) and the DK software package license file, respectively.

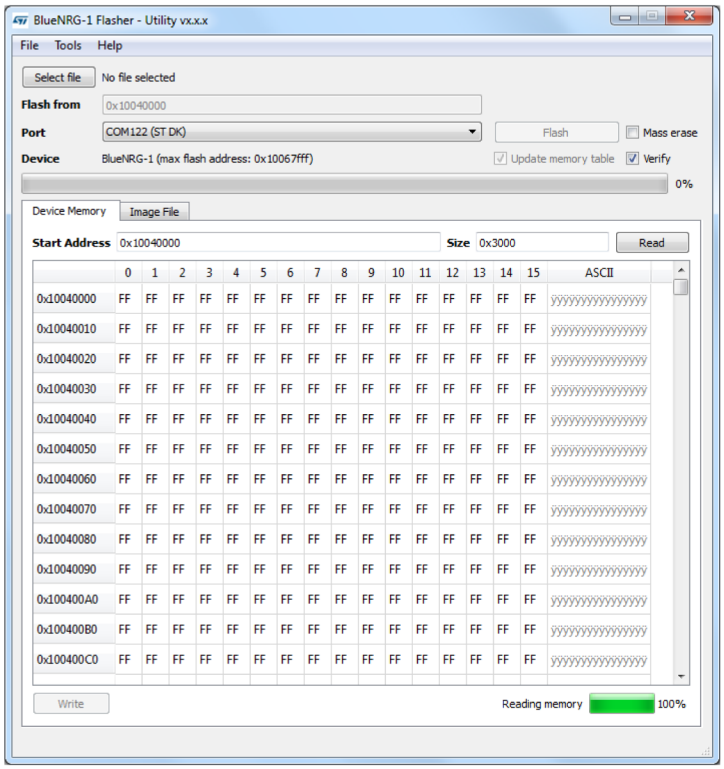
4 BlueNRG-1 Flasher utility

The BlueNRG-1 Flasher utility allows BlueNRG-1, BlueNRG-2 programming using the UART bootloader.

4.1 How to run

User can run this utility by clicking on the BlueNRG-1 Flasher icon under: Start → STMicroelectronics → BlueNRG -1_2 DK X.X.X → BlueNRG1 Flasher or Start → STMicroelectronics → BlueNRG -1_V1 DK X.X.X → BlueNRG1 Flasher

Figure 15. BlueNRG-1 Flasher utility



4.2 Main user interface window

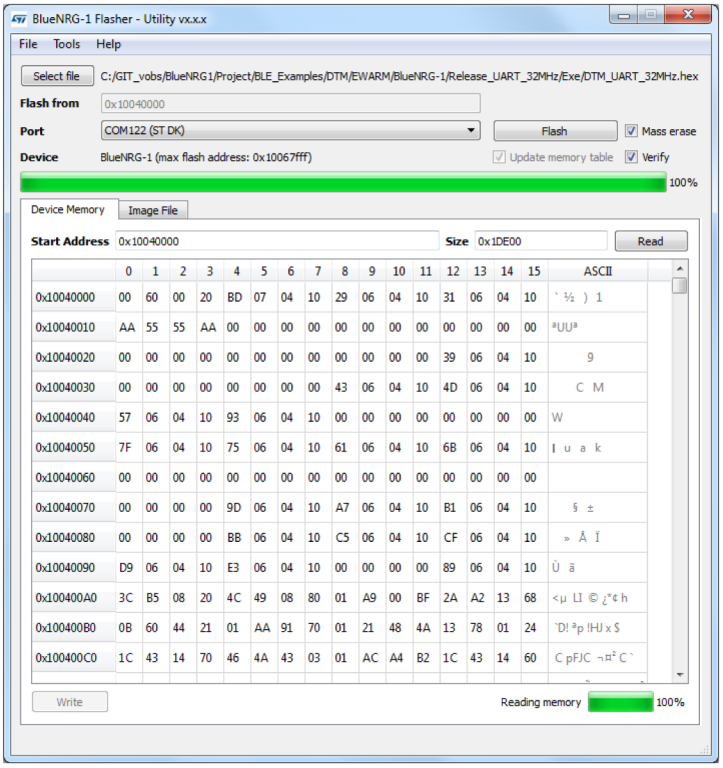
In the upper section of the BlueNRG-1 Flasher – Utility main window, you can:

- select the image file (‘Select file’ button)
- choose the flashing address (‘Flash from’ text input bar, only enabled for .bin files)
- select the COM port to be used to interface the device (‘Port’ dropdown list)

The BlueNRG-1, BlueNRG-2 device memory is read when the associated COM port is opened.

The serial baudrate used for BlueNRG-1, BlueNRG-2 evaluation board is 460800 bps.

Figure 16. **BlueNRG-1 Flasher utility main window**



4.2.1 Main menu items

From the ‘File’ menu, you can:

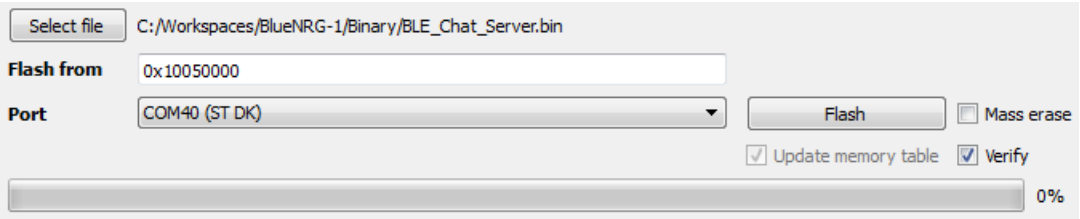
- Load an existing .bin or .hex (Intel extended) file.
- Save the current memory image in a .bin file. The start address and the size of the memory section to be saved to file are selectable from the ‘Device Memory’ tab.
- Close the application.

From the ‘Tools’ menu, you can mass erase all the device Flash memory.

4.2.2 Image file selection

Use the ‘Select file’ button on the main page (or the File>Load menu) to load an existing .bin or .hex file. The full path of the selected file appears next to the button and the ‘Flash’ becomes active.

Figure 17. **BlueNRG-1 Flasher utility file selection**



By default, the ‘Mass erase’ option beside the ‘Flash’ button is not checked, and only the required memory pages are erased and written with the file content. When this option is checked, the memory flash phase is preceded by a full mass erase.

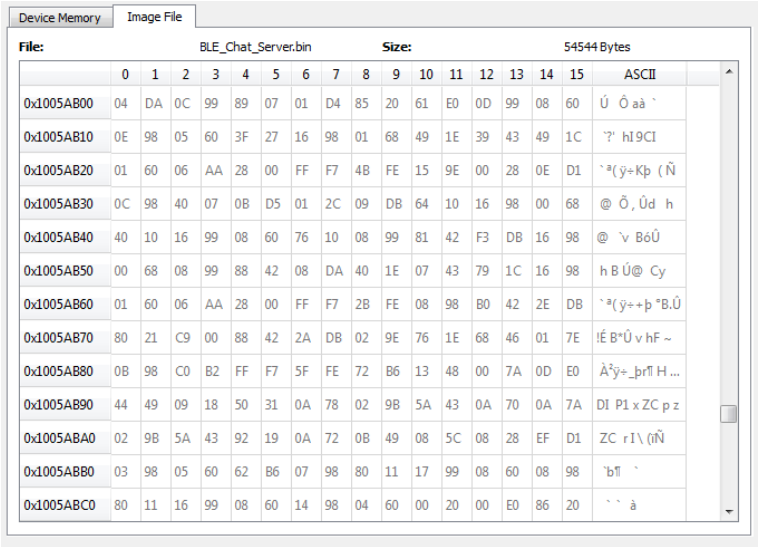
The ‘Verify’ option forces a check to ensure that the memory content has been written correctly.

Check the ‘Update memory table’ option to update the ‘Device Memory’ table after the flashing operation. This option is automatically checked when the ‘Verify’ checkbox is selected.

4.2.3 ‘Image File’ tab

The selected file name, size and parsed contents to be flashed to device memory can be viewed in the ‘Image File’ tab.

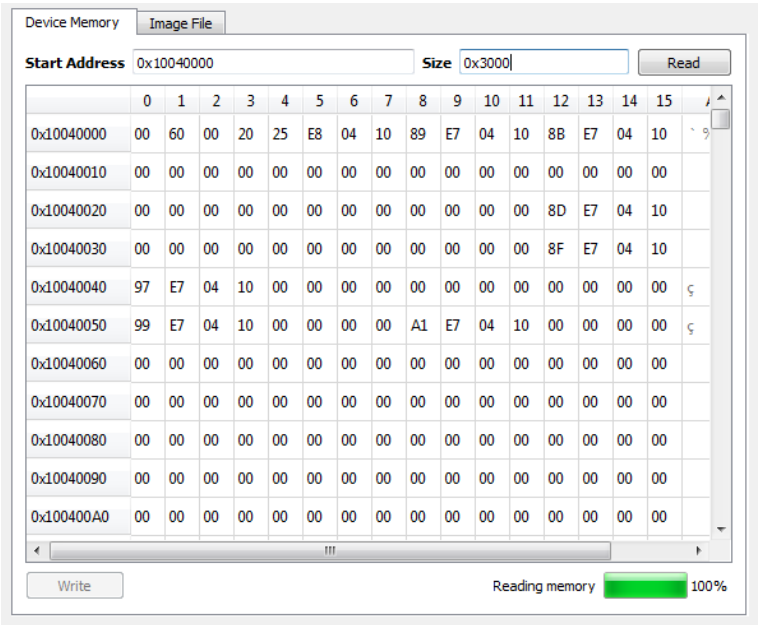
Figure 18. BlueNRG-1 Flasher utility image file viewer



4.2.4 ‘Device Memory’ tab

Select this tab to view the memory contents of a connected device.

Figure 19. BlueNRG-1 Flasher utility device memory viewer



Click the ‘Read’ button to transfer the memory segment defined by ‘Start Address and ‘Size’ into the table.

The first column gives the base address of the following 16 bytes in a row (e.g., row 0x10040050, column 4 holds the hexadecimal byte value at 0x10040054.

You can change byte values by double-clicking a cell and entering a new hexadecimal value; edited bytes appear in red.

Click the ‘Write’ button to flash the entire page with the new byte values into device memory.

Figure 20. BlueNRG-1 Flasher utility changing memory fields

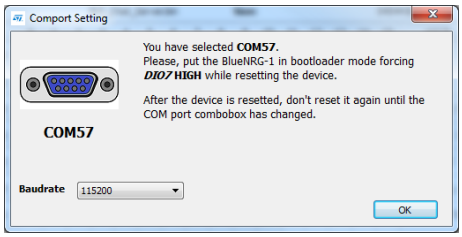
0x10040040	97	E7	04	10	00	00	00	00	00	00	00	0(
0x10040050	99	E7	04	14	00	00	00	00	A1	E7	04	1(
0x10040060	00	00	00	00	00	00	00	00	00	00	00	0(

4.2.5 Using BlueNRG-1 Flasher utility with other boards

The BlueNRG-1 Flasher utility automatically detects BlueNRG-1, BlueNRG-2 evaluation boards like STEVAL-IDB007Vx, STEVAL-IDB008Vx and uses an auxiliary STM32 to (driven by the GUI) to reset the BlueNRG-1, BlueNRG-2 and put it into bootloader mode.

The application also works with custom boards providing simple UART access to the BlueNRG-1, BlueNRG-2 device, but you must put the device in bootloader mode manually. Upon the selection of any non-STEVAL COM port, the following popup appears.

Figure 21. BlueNRG-1 Flasher utility ‘Comport Setting’ popup



When this popup appears, set the BlueNRG-1, BlueNRG-2 pin DIO7 high and reset the BlueNRG-1, BlueNRG-2 device (keeping the DIO7 high); the device should now be in bootloader mode.

You can also set a preferred Baudrate for the UART in the popup window and then press OK to return to the GUI.

Note: Avoid resetting the device while using the BlueNRG-1 Flasher utility unless the Comport Setting popup is active. If the device is reset, you must toggle the COM port to use the Flasher utility again.

5 BlueNRG-1 radio parameters wizard

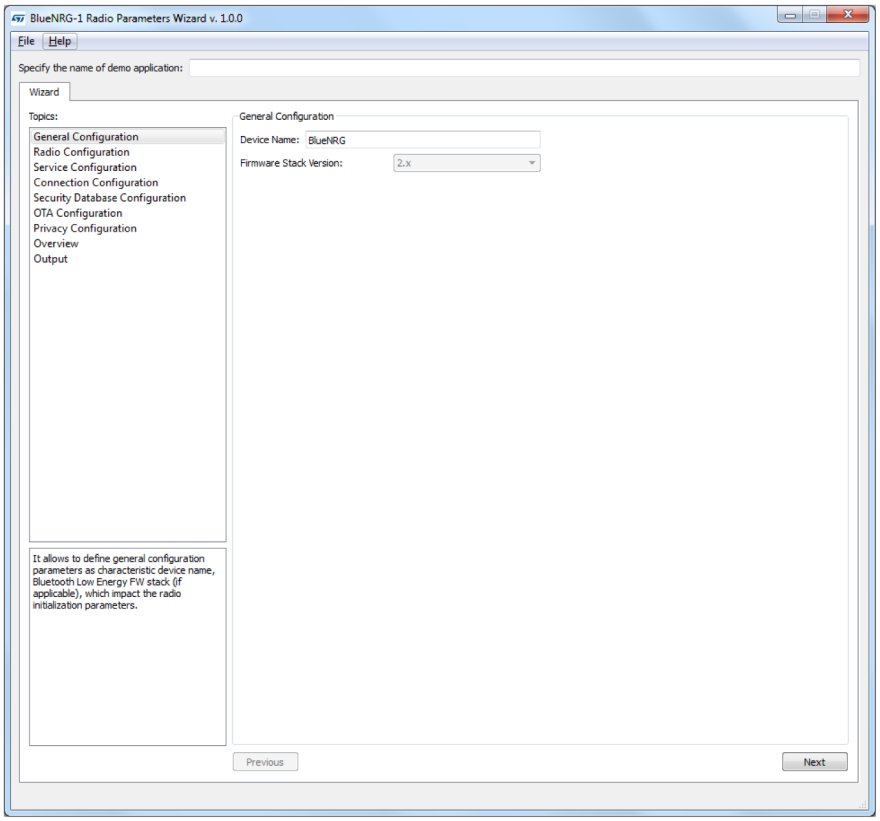
The BlueNRG-1 Radio Parameters Wizard is a PC application which allows to define the proper values required for the correct BlueNRG-1, BlueNRG-2 BLE radio initialization, based on the specific user application scenario. As consequence of the user choices, a configuration header file (*_config.h) is generated: this file must be used on the user demonstration application folder.

Note: The BlueNRG-1 radio parameters wizard is provided only on BlueNRG-1_2 DK SW package (STSW-BLUENRG1-DK) supporting BLE stack v2.x family.

5.1 How to run

User can run this utility by clicking on the BlueNRG-1 Radio Init Wizard icon under: Start → STMicroelectronics → BlueNRG -1_2 DK X.X.X

Figure 22. BlueNRG-1 radio parameters wizard



5.2 Main user interface window

In the left section of the BlueNRG-1 Radio Initialization Parameters Wizard Utility, user can select the following topics allowing to define the specific radio initialization parameters based on the specific BLE application requirements:

1. General Configuration
2. Radio Configuration
3. Service Configuration
4. Connection Configuration

5. Security DataBase configuration
6. OTA configuration
7. Privacy configuration
8. Overview
9. Output

Refer to the BlueNRG-1 Radio Parameters Wizard documentation available within BlueNRG-1_2 DK SW package for more details about each provided configuration section.

6 Programming with BlueNRG-1, BlueNRG-2 system on chip

The BlueNRG-1, BlueNRG-2 Bluetooth low energy (BLE) stack is provided as a binary library. A set of APIs to control BLE functionality. Some callbacks are also provided for user applications to handle BLE stack events. The user is simply requested to link this binary library to his or her application and use the relevant APIs to access BLE functions and complete the stack event callbacks to manage responses according to application requirements.

A set of software driver APIs is also included for accessing the BlueNRG-1, BlueNRG-2 SoC peripherals and resources (ADC, GPIO, I²C, MFTX, Micro, RTC, SPI, SysTick, UART and WDG).

The development kit software includes sample code demonstrating how to configure BlueNRG-1, BlueNRG-2 and use the device peripherals and BLE APIs and event callbacks. Documentation on the BLE APIs, callbacks, and peripheral drivers are provided in separate documents.

6.1 Software directory structure

The BlueNRG-1, BlueNRG-2 DK software packages files are organized in the following main directories:

- **Application**: containing BlueNRG-1, BlueNRG-2 Navigator, Flasher and Radio Parameters Wizard PC applications.
- **Doc**: with doxygen BLE APIs and events, BlueNRG-1, BlueNRG-2 peripheral drivers, BLE demo applications, BlueNRG-1, BlueNRG-2 Peripheral examples, BlueNRG-1, BlueNRG-2 SDK and HAL driver documentation, DK release notes and license file.
- **Firmware**: with prebuilt binary BLE and peripheral driver sample applications.
- **Library**
  - **Bluetooth LE**: Bluetooth low energy stack binary library and all the definitions of stack APIs, stack events callbacks and constants. Over-the-air Bluetooth low energy firmware upgrade support if applicable.
  - **BlueNRG1_Periph_Driver**: BlueNRG-1, BlueNRG-2 drivers for device peripherals (ADC, clock, DMA, Flash, GPIO, I²C, timers, RTC, SPI, UARR and watchdog).
  - **CMSIS**: BlueNRG-1 CMSIS files.
  - **SDK_Eval_BlueNRG1**: SDK drivers providing an API interface to the BlueNRG-1, BlueNRG-2 platform hardware resources (LEDs, buttons, sensors, I/O channel).
  - **HAL**: Hardware abstraction level APIs for abstracting certain BlueNRG-1 hardware features (sleep modes, clock based on SysTick, etc.).
- **Project**
  - **BLE_Examples**: Bluetooth low energy demonstration application including Headers, source files and EWARM, Keil and Atollic project files.
  - **BlueNRG1_Periph_Examples**: with sample applications for the BlueNRG-1, BlueNRG-2 peripherals and hardware resources, including Headers, source files and project files.
- **Utility**: contains some utilities

Note: The selection between BlueNRG-1, BlueNRG-2 device is done at compile time using a specific define value BLUENRG2_DEVICE for selecting BlueNRG-2 device. Default configuration (no define value) selects BlueNRG-1 device.

Note: BlueNRG-1_V1 DK SW package framework refers only BlueNRG-1 device.

7 BLE beacon demonstration application

The BLE beacon demo is supported by the BlueNRG-1,BlueNRG-2 development platforms (STEVAL-IDB007Vx, STEVAL-IDB008Vx). It demonstrates how to configure a BlueNRG-1 device to advertise specific manufacturing data and allow another BLE device to determine whether it is in BLE beacon device range.

7.1 BLE Beacon application setup

This section describes how to configure a BLE device to act as a beacon device.

7.1.1 Initialization

The BLE stack must be correctly initialized thus:

```
aci_gatt_init();
```

```
aci_gap_init(GAP_PERIPHERAL_ROLE, 0, 0x08, &service_handle, &dev_name_char_handle,
 &appearance_char_handle);
```

See the BLE stack documentation for more information on these and following commands.

7.1.2 Define advertising data

The BLE Beacon application advertises the following manufacturing data:

Table 5. BlueNRG-1 Beacon advertising manufacturing data

Data field	Description	Notes
Company identifier code	SIG company identifier (1)	Default is 0x0030 (STMicroelectronics)
ID	Beacon ID	Fixed value
Location UUID	Beacons UUID	Used to distinguish specific beacons from others
Major number	Identifier for a group of beacons	Used to group a related set of beacons
Minor number	Identifier for a single beacon	Used to identify a single beacon
Tx Power	2's complement of the Tx power	Used to establish how far you are from device

1. *available at: https://www.bluetooth.org/en-us/specification/assigned-numbers/company-identifiers*

7.1.3 Entering non-connectable mode

The BLE Beacon device uses the GAP API command to enter non-connectable mode thus:

```
aci_gap_set_discoverable(ADV_NONCONN_IND, 160, 160, PUBLIC_ADDR,
```

```
                         NO_WHITE_LIST_USE,0, NULL, 0, NULL, 0, 0);
```

To advertise the specific selected manufacturer data, the BLE Beacon application can use the following GAP APIs:

```
/* Remove TX power level field from the advertising data: it is necessary to have
```

```
enough space for the beacon manufacturing data */
```

```
aci_gap_delete_ad_type(AD_TYPE_TX_POWER_LEVEL);
```

```
/* Define the beacon manufacturing payload */

uint8_t manuf_data[] = {26, AD_TYPE_MANUFACTURER_SPECIFIC_DATA, 0x30,   0x00, //Co
mpany identifier code (Default is 0x0030 - STMicroelectronics) 0x02,// ID

0x15,//Length of the remaining payload

0xE2, 0x0A, 0x39, 0xF4, 0x73, 0xF5, 0x4B, 0xC4, //Location UUID

0xA1, 0x2F, 0x17, 0xD1, 0xAD, 0x07, 0xA9, 0x61,

0x00, 0x02, // Major number

0x00, 0x02, // Minor number

0xC8//2's complement of the Tx power (-56dB)};

};

/* Set the beacon manufacturing data on the advertising packet */ aci_gap_update_a
dv_data(27, manuf_data);
```

8 BLE chat demo application

The BLE chat demo (server and client roles) is supported on the BlueNRG-1, BlueNRG-2 development platforms (STEVAL-IDB007Vx, STEVAL-IDB008Vx). It implements simple two-way communication between two BLE devices, demonstrating point-to-point wireless communication using the BlueNRG-1 product.

This demo application exposes a single chat service with the following (20 byte max.) characteristic values:

- The TX characteristic, with which the client can enable notifications; when the server has data to be sent, it sends notifications with the value of the TX characteristic.
- The RX characteristic, is a writable characteristic; when the client has data to be sent to the server, it writes a value in this characteristic.

There are two device roles which can be selected through the specific project workspace:

- The Server that exposes the chat service (BLE peripheral device).
- The Client that uses the chat service (BLE central device).

The application requires two devices to be programmed with respective server and client roles. These must be connected to a PC via USB with an open serial terminal for each device, with the following configurations:

Table 6. Serial port configuration

Parameter	value
Baudrate	115200 bit/s
Data bits	8
Parity bits	None
Stop bits	1

The application listens for keys typed in one device terminal and sends them to the remote device when the return key is pressed; the remote device then outputs the received RF messages to the serial port. Therefore, anything typed in one terminal becomes visible in the other.

8.1 Peripheral and central device setup

This section describes how two BLE chat devices (server-peripheral and client-central) interact with each other in order to set up a point-to-point wireless chat.

BLE device must first be set up on both devices by sending a series of API commands to the processor.

8.1.1 Initialization

The BLE stack must be correctly initialized before establishing a connection with another BLE device. This is done with `aci_gatt_init()` and `aci_gap_init()` APIs:

```
aci_gatt_init();
```

BLE Chat server role:

```
aci_gap_init(GAP_PERIPHERAL_ROLE, 0, 0x08, &service_handle, &dev_name_char_handle,
 &appearance_char_handle);
```

BLE Chat client role:

```
aci_gap_init(GAP_CENTRAL_ROLE, 0, 0x08, &service_handle, &dev_name_char_handle, &a
ppearance_char_handle);
```

Peripheral and central BLE roles must be specified in the `aci_gap_init()` command. See the BLE stack API documentation for more information on these and following commands.

8.1.2 Add service and characteristics

The chat service is added to the BLE chat server device via:

```
aci_gatt_add_service(UUID_TYPE_128, &service_uuid, PRIMARY_SERVICE, 7,&chatServHan
dle);
```

Where `service_uuid` is the private service 128-bit UUID allocated for the chat service (Primary service). The command returns the service handle in `chatServHandle`. The TX characteristic is added using the following command on the BLE Chat server device:

```
aci_gatt_add_char(chatServHandle, UUID_TYPE_128, &charUuidTX, 20, CHAR_PROP_NOTIFY
, ATTR_PERMISSION_NONE, 0, 16, 1, &TXCharHandle);
```

Where `charUuidTX` is the private characteristic 128-bit UUID allocated for the TX characteristic (notify property). The characteristic handle is returned on the `TXCharHandle` variable.

The RX characteristic is added using the following command on the BLE Chat server device:

```
aci_gatt_add_char(chatServHandle, UUID_TYPE_128, &charUuidRX, 20, CHAR_PROP_WRITE|
CHAR_PROP_WRITE_WITHOUT_RESP, ATTR_PERMISSION_NONE, GATT_SERVER_ATTR_WRITE,16, 1,
&RXCharHandle);
```

Where `charUuidRX` is the private characteristic 128-bit UUID allocated for the RX characteristic (write property). The characteristic handle is returned on the `RXCharHandle` variable.

See the BLE stack API documentation for more information on these and following commands.

8.1.3 Enter connectable mode

The server device uses GAP API commands to enter the general discoverable mode:

```
aci_gap_set_discoverable(ADV_IND, 0, 0, PUBLIC_ADDR, NO_WHITE_LIST_USE,8,local_nam
e, 0, NULL, 0, 0);
```

The `local_name` parameter contains the name presented in advertising data, as per Bluetooth core specification version 4.2, Vol. 3, Part C, Ch. 11.

8.1.4 Connection with central device

Once the server device is discoverable by the BLE chat client device, the client device uses `aci_gap_create_connection()` to connect with the BLE chat server device:

```
aci_gap_create_connection(0x4000, 0x4000, PUBLIC_ADDR, bdaddr, PUBLIC_ADDR, 40, 40
, 0, 60, 2000 , 2000);
```

Where `bdaddr` is the peer address of the client device.

Once the two devices are connected, you can set up corresponding serial terminals and type messages in either of them. The typed characters are stored in two respective buffers and when the return key is pressed:

- on the BLE chat server device, the typed characters are sent to the BLE chat client device by notifying the previously added TX characteristic (after notifications are enabled) with:

```
aci_gatt_update_char_value(chatServHandle,TXCharHandle,0,len, (uint8_t*)cmd+j);
```

- on the BLE chat client device, the typed characters are sent to the BLE chat server device by writing the previously added RX characteristic with:

```
aci_gatt_write_without_resp(connection_handle, rx_handle+1, len, (uint8_t *)cmd+j)
;
```

Where `connection_handle` is the handle returned upon connection as a parameter of the connection complete event, rx_handle is the RX characteristic handle discovered by the client device.

Once these API commands have been sent, the values of the TX and RX characteristics are displayed on the serial terminals.

Figure 23. BLE chat client

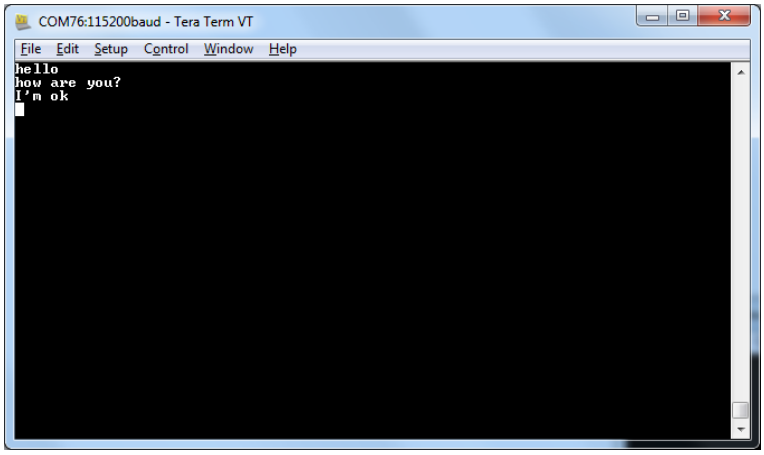
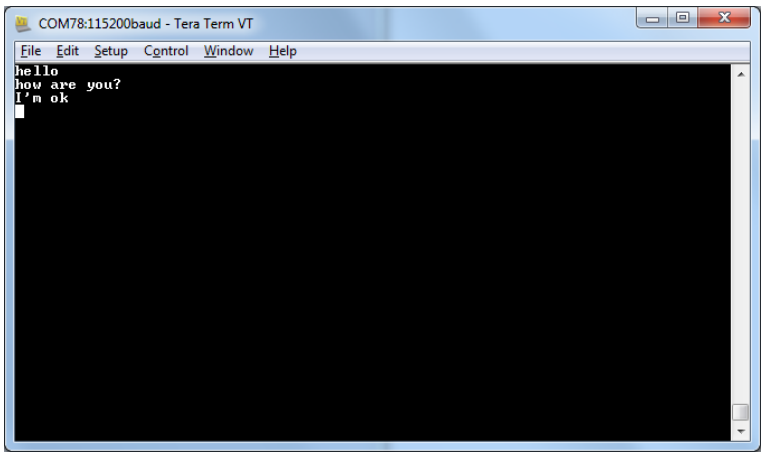


Figure 24. BLE chat server



9 BLE chat master and slave demo application

The BLE chat master and slave demo is supported on the BlueNRG-1, BlueNRG-2 development platforms (STEVAL-IDB007Vx, STEVAL-IDB008Vx). It demonstrates simple point-to-point wireless communication using a single application which configures the chat client and server roles at runtime.

The new chat demo application configures a BLE device as central or peripheral using the API:

```
aci_gap_init(GAP_CENTRAL_ROLE|GAP_PERIPHERAL_ROLE, 0, 0x07, &service_handle, &dev_
name_char_handle, &appearance_char_handle);
```

It then initiates a discovery procedure for another BLE device configured with the same chat master and slave application image.

If such a device is found within a random interval, it starts a connection procedure and waits until a connection is established. If the discovery procedure time expires without finding another chat master and slave device, the device enters discovery mode and waits for another chat master and slave device to discover and connect to it.

When connection is established, the client and server roles are defined and the chat communication channel can be used.

This demo application exposes a single chat service with the following (20 byte max.) characteristic values:

- The TX characteristic, with which the client can enable notifications; when the server has data to be sent, it sends notifications with the value of the TX characteristic.
- The RX characteristic, is a writable characteristic; when the client has data to be sent to the server, it writes a value in this characteristic.

The application requires two devices to be programmed with the same application, with the server and client roles defined at runtime. Connect the two devices to a PC via USB and open a serial terminal on both with the same configuration as Table 6. Serial port configuration.

The application listens for keys typed in one device terminal and sends them to the remote device when the return key is pressed; the remote device then outputs the received RF messages to the serial port. Therefore, anything typed in one terminal becomes visible in the other.

9.1 BLE chat master and slave roles

This section describes how two BLE chat master and slave devices interact with each other in order to set up a point-to-point wireless chat.

The BLE stack must first be set up on both devices by sending a series of API commands to the processor. The chat master and slave client and server roles are defined at runtime.

9.1.1 Initialization

The BLE stack must be correctly initialized before establishing a connection with another BLE device. This is done with two commands:

```
aci_gatt_init();
```

```
aci_gap_init(GAP_CENTRAL_ROLE|GAP_PERIPHERAL_ROLE, TRUE,0x07, &service_handle, &de
v_name_char_handle, &appearance_char_handle);
```

The BLE peripheral and central roles are specified in the `aci_gap_init()` command. See the BLE API documentation for more information on these and following commands.

9.1.2 Add service and characteristics

Refer to Section 8.1.2 Add service and characteristics .

9.1.3 Start discovery procedure

To find another BLE chat master and slave device in discovery mode, a discovery procedure must be started via:

```
aci_gap_start_general_discovery_proc(0x4000, 0x4000, 0x00, 0x00);
```

9.1.4 Enter connectable mode

The following GAP API command is used for entering general discoverable mode:

```
aci_gap_set_discoverable(ADV_IND, 0x90, 0x90, PUBLIC_ADDR, NO_WHITE_LIST_USE, size
of(local_name), local_name, 0, NULL, 0x6, 0x8);
```

9.1.5 Connection with chat master and slave client device

In the above mentioned discovery and mode assignment procedures, the two chat master and slave applications assume respective client and server roles at runtime. During this initial configuration phase, when a chat master and slave device is placed in discoverable mode and it is found by the other chat master and slave device performing a discovery procedure, a Bluetooth low energy connection is created and the device roles are defined.

The following GAP API command is used for connecting to the discovered device:

```
aci_gap_create_connection(0x4000, 0x4000,device_found_address_type, device_found_a
ddress, PUBLIC_ADDR, 40, 40, 0, 60, 2000 , 2000);
```

Where `device_found_address_type` is the address type of the discovered chat master and slave and `device_found_address` is the peer address of the discovered chat master and slave device.

Once the two devices are connected, you can set up corresponding serial terminals and type messages in either of them. The typed characters are stored in two respective buffers and when the return key is pressed:

On the BLE chat master-and-slave server device, the typed characters are sent to the master-and-slave client device by notifying the previously added TX characteristic (after notifications have been enabled). This is done via:

```
aci_gatt_update_char_value(chatServHandle, TXCharHandle, 0, len, (uint8_t *)cmd+j)
;
```

On the master-and-slave client device, the typed characters are sent to the master-and-slave server device, by writing the previously added RX characteristic. This is done via:

```
aci_gatt_write_without_resp (connection_handle, rx_handle +1, len, (uint8_t *)cmd+
j);
```

Where `connection_handle` is the handle returned upon connection as a parameter of the connection complete event, `rx_handle` is the RX characteristic handle discovered by the client device.

Once these API commands have been sent, the values of the TX and RX characteristics are displayed on the serial terminals.

10 BLE remote control demo application

The BLE remote control application is supported on the BlueNRG-1, BlueNRG-2 development platforms (STEVAL-IDB007Vx, STEVAL-IDB008Vx). It demonstrates how to control a remote device (like an actuator) using a BlueNRG-1, BlueNRG-2 device.

This application periodically broadcasts temperature values that can be read by any device. The data is encapsulated in a manufacturer-specific AD type and the content (besides the manufacturer ID, i.e., 0x0030 for STMicroelectronics) is as follows:

Table 7. BLE remote advertising data

Byte 0	Byte 1	Byte2
App ID (0x05)	Temperature value (little-endian)	

The temperature value is given in tenths of degrees Celsius.

The device is also connectable and exposes a characteristic used to control LEDs DL1, DL2 and DL3 on the BLE kit platform. The value of this characteristic is a bitmap of 1 byte. Each bit controls one of the LEDs:

- bit 0 is the status of LED DL1
- bit 1 is the status of LED DL2
- bit 2 is the status of LED DL3.

A remote device can therefore connect and write this byte to change or read the status of these LEDs (1 for LED ON, 0 for LED OFF).

The peripheral disconnects after a timeout (`DISCONNECT_TIMEOUT`) to prevent a central device remaining connected to the device indefinitely.

Security is not enabled by default, but this can be changed with `ENABLE_SECURITY` (refer to file BLE_RC_main.h). When security is enabled, the central device must be authenticated before reading or writing the device characteristic.

To interact with a device configured as a BLE remote control, another BLE device (a BlueNRG-1, BlueNRG-2 or any Bluetooth® smart ready device) can be used to detect and view broadcast data.

To control one of the LEDs, the device has to connect to a BlueNRG-1 BLE remote control device and write in the exposed control point characteristic. The Service UUID is ed0ef62e-9b0d-11e4-89d3-123b93f75cba. The control point characteristic UUID is ed0efb1a-9b0d-11e4-89d3-123b93f75cba.

10.1 BLE remote control application setup

This section describes how to configure a BlueNRG-1 device to acting as a remote control device.

10.1.1 Initialization

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

```
aci_gatt_init();
```

```
aci_gap_init(GAP_PERIPHERAL_ROLE, 0, 0x07, &service_handle, &dev_name_char_handle,
 &appearance_char_handle);
```

See BLE stack API documentation for more information on these and following commands.

10.1.2 Define advertising data

The BLE remote control application advertises certain manufacturing data as follows:

```
/* Set advertising device name as Node */
const uint8_t scan_resp_data[] = {0x05,AD_TYPE_COMPLETE_LOCAL_NAME,'N','o','d','e'
}
/* Set scan response data */   hci_le_set_scan_response_data(sizeof(scan_resp_data
),scan_resp_data);
/* Set Undirected Connectable Mode */
aci_gap_set_discoverable(ADV_IND, (ADV_INTERVAL_MIN_MS*1000)/625, (ADV_INTERVAL_MA
X_MS*1000)/625, PUBLIC_ADDR, NO_WHITE_LIST_USE, 0, NULL, 0, NULL, 0, 0);
/* Set advertising data */
hci_le_set_advertising_data(sizeof(adv_data),adv_data);
```

On the development platform, the temperature sensor value is set in the `adv_data` variable.

10.1.3 Add service and characteristics

The BLE Remote Control service is added via:

```
aci_gatt_add_service(UUID_TYPE_128, &service_uuid, PRIMARY_SERVICE, 7, &RCServHand
le);
```

Where `service_uuid` is the private service 128-bit UUID allocated for the BLE remote service (ed0ef62e-9b0d-11e4-89d3-123b93f75cba).

The command returns the service handle in `RCServHandle`.

The BLE remote control characteristic is added using the following command:

```
#if ENABLE_SECURITY
```

```
aci_gatt_add_char(RCServHandle, UUID_TYPE_128, &controlPointUuid, 1, CHAR_PROP_REA
D|CHAR_PROP_WRITE|CHAR_PROP_WRITE_WITHOUT_RESP|CH AR_PROP_SIGNED_WRITE, ATTR_PERMI
SSION_AUTHEN_READ|ATTR_PERMISSION_AUTHEN_WRITE, GATT_NOTIFY_ATTRIBUTE_WRITE,16,1,&
controlPointHandle);
```

```
#else
```

```
aci_gatt_add_char(RCServHandle, UUID_TYPE_128, &controlPointUuid, 1, CHAR_PROP_REA
D|CHAR_PROP_WRITE|CHAR_PROP_WRITE_WITHOUT_RESP, ATTR_PERMISSION_NONE, GATT_NOTIFY_
ATTRIBUTE_WRITE, 16, 1,&controlPointHandle);
```

```
#endif
```

Where `controlPointUuid` is the private characteristic 128-bit UUID allocated for BLE remote control characteristic (ed0efb1a-9b0d-11e4-89d3-123b93f75cba) and `controlPointHandle` is the BLE remote control characteristic handle.

If security is enabled, the characteristic properties must be set accordingly to enable authentication on `controlPointUuid` characteristic read and write.

10.1.4 Connection with a BLE Central device

When connected to a BLE central device (another BlueNRG-1, BlueNRG-2 device or any Bluetooth® smart ready device), the `controlPointUuid` characteristic is used to control the BLE remote control platform LED. Each time a write operation is performed on `controlPointUuid`, the `aci_gatt_attribute_modified_event()` callback is raised and the selected LEDs are turned on or off.

11 BLE sensor profile demo

The BLE sensor profile demo is supported on the BlueNRG-1, BlueNRG-2 development platforms (STEVAL-IDB007Vx, STEVAL-IDB008Vx). It implements a proprietary, Bluetooth low energy (BLE) sensor profile.

This example is useful for building new profiles and applications that use the BlueNRG-1, BlueNRG-2 SoC. The GATT profile is not compliant with any existing specifications as the purpose of this project is to simply demonstrate how to implement a given profile.

This profile exposes the acceleration and environmental services.

Figure 25. BLE sensor demo GATT database shows the whole GATT database, including the GATT (0x1801) and GAP (0x1800) services that are automatically added by the stack.

The acceleration service's free fall characteristic cannot be read or written, but can be signaled. The application sends notification of this characteristic (with a value of 0x01) if a free fall condition is detected by the MEMS sensor (when the acceleration on the three axes is near zero for a certain amount of time). Notifications can be enabled or disabled by writing the associated client characteristic configuration descriptor.

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

Another service is defined, which contains characteristics that expose data from some environmental sensors: temperature and pressure. Each characteristic data type is described in a format descriptor. All of the characteristics have read-only properties.

Figure 25. BLE sensor demo GATT database

# Handle	UUID (16 or 128bit)	Attribute Type	Properties	Initial Parameter Value	Comment
			B R D / R D / W R E / N O T / S I G N / E X T		
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	
12 000C	2800	Primary Service		{Service=0x02366E80CF3A11E19AB40002A5D5C51B ("Acc Service")}	
13 000D	2803	Characteristic	X	{handle=0x000E, UUID=0xE23E78A0CF4A11E18FFC0002A5D5C51B}	
14 000E	E23E78A0CF4A11E18FFC0002A5D5C51B	**Free Fall**		0x00	Indication with value 1 when a freel fall condition is detected
15 000F	2902	Client Characteristic Configuration		0x0000	
16 0010	2803	Characteristic	X X	{handle=0x0011, UUID=0x340A1B80CF4B11E1AC360002A5D5C51B}	
17 0011	340A1B80CF4B11E1AC360002A5D5C51B	**Acceleration**		0x000000000000	X-Axis (2bytes) Y-Axis (2bytes) Z-Axis (2bytes)
18 0012	2902	Client Characteristic Configuration		0x0000	
19 0013	2800	Primary Service		{Service=0x42821A40E47711E282D00002A5D5C51B ("Env Service")}	
20 0014	2803	Characteristic	X	{handle=0x0015, UUID=0xA32E5520E47711E2A9E30002A5D5C51B}	
21 0015	A32E5520E47711E2A9E30002A5D5C51B	**Temperature**		0x0000	Temperature in tenths of degree Celsius
22 0016	2904	Characteristic Format		{format=0x0E, exp=-1, unit=0x272F, n_sp=0x00, descr=0x0000}	format=sint16, unit=temperature celsius
23 0017	2803	Characteristic	X	{handle=0x0018, UUID=0xCD20C480E48B11E2840B0002A5D5C51B}	
24 0018	CD20C480E48B11E2840B0002A5D5C51B	**Pressure**		0x000000	Pressure in hundredths of millibar
25 0019	2904	Characteristic Format		{format=0x0F, exp=-5, unit=0x2780, n_sp=0x00, descr=0x0000}	format=sint24, unit=pressure bar
26 001A	2803	Characteristic	X	{handle=0x001B, UUID=0x01C50B60E48C11E2A0730002A5D5C51B}	

11.1 BlueNRG app for smartphones

An application is available for iOS™ and Android™ smartphones or tablets that also works with the BLE sensor profile demo. This app enables notification of the acceleration characteristic and displays the value on screen. Data from environmental sensors are also periodically read and displayed.

Figure 26. BlueNRG sensor app

11.2 BLE sensor profile demo: connection with a central device

This section describes how to interact with a central device, while the BLE stack is acting as a peripheral. The central device may be another BlueNRG-1, BlueNRG-2 device acting as a BLE master, or any other Bluetooth smart or Bluetooth smart ready device.

The BLE stack must first be set up by sending a series of BLE API commands to the processor.

11.2.1 Initialization

The BLE stack must be correctly initialized before establishing a connection with another Bluetooth LE device. This is done via:

```
aci_gatt_init();
```

```
aci_gap_init(GAP_PERIPHERAL_ROLE, 0, 0x07, &service_handle, &dev_name_char_handle,
 &appearance_char_handle);
```

See BLE stack API documentation for more information on these and following commands.

11.2.2 Add service and characteristics

The BlueNRG-1 BLE stack has both server and client capabilities. A characteristic is an element in the server database where data is exposed, while a service contains one or more characteristics. The acceleration service is added with the following command:

```
aci_gatt_add_service(UUID_TYPE_128,  &service_uuid, PRIMARY_SERVICE, 7,  &accServH
andle);
```

The command returns the service handle on variable `accServHandle`. The free fall and acceleration characteristics must now be added to this service thus:

```
aci_gatt_add_char(accServHandle, UUID_TYPE_128, &char_uuid, 1, CHAR_PROP_NOTIFY, A
TTR_PERMISSION_NONE, 16, 0, &freeFallCharHandle);
```

```
aci_gatt_add_char(accServHandle, UUID_TYPE_128, &char_uuid, 6, CHAR_PROP_NOTIFY|CH
AR_PROP_READ, ATTR_PERMISSION_NONE, GATT_NOTIFY_READ_REQ_AND_WAIT_FOR_APPL_RESP, 1
6, 0, &accCharHandle);
```

The free fall and acceleration characteristics handles are returned on `freeFallCharHandle` and `accCharHandle` variables respectively.

Similar steps are followed for adding the environmental sensor and relative characteristics.

11.2.3 Enter connectable mode

Use GAP API command to enter one of the discoverable and connectable modes:

```
aci_gap_set_discoverable(ADV_IND, (ADV_INTERVAL_MIN_MS*1000)/625, (ADV_INTERVAL_MA
X_MS*1000)/625, STATIC_RANDOM_ADDR, NO_WHITE_LIST_USE sizeof(local_name), local_na
me, 0, NULL, 0, 0);
```

Where

```
local_name[] = {AD_TYPE_COMPLETE_LOCAL_NAME,'B','l','u','e','N','R','G'};
```

The `local_name` parameter contains the name presented in advertising data, as per Bluetooth core specification version, Vol. 3, Part C, Ch. 11.

11.2.4 Connection with central device

Once the BLE stack is placed in discoverable mode, it can be detected by a central device. The smartphone app described in Section 11.1 BlueNRG app for smartphones is designed for interact with the sensor profile demos (it also supports the BlueNRG-1 device).

Any Bluetooth smart or Bluetooth smart ready device like a smartphone can connect to the BLE sensor profile demo.

For example, the LightBlue application in Apple Store® connects iPhone® versions 4S/5 and above can connect to the sensor profile device. When you use the LightBlue application, detected devices appear on the screen with the BlueNRG name. By tapping on the box to connect to the device, a list of all the available services is shown on the screen; tapping a service shows the characteristics for that service.

The acceleration characteristic can be notified using the following command:

```
aci_gatt_update_char_value(accServHandle, accCharHandle, 0, 6, buff);
```

Where `buff` is a variable containing the three axes acceleration values.

Once this API command has been sent, the new value of the characteristic is displayed on the phone.

12 BLE sensor profile center demo

The BLE sensor profile central demo is supported on the BlueNRG-1, BlueNRG-2 development platforms (STEVAL-IDB007Vx, STEVAL-IDB008Vx). It implements a basic version of the BLE Sensor Profile Central role which emulates the Sensor Demo applications available for smartphones (iOS and android).

This application configures a BlueNRG-1, BlueNRG-2 device as a Sensor device, Central role which is able to find, connect and properly configure the free fall, acceleration and environment sensors characteristics provided by a BLE development platform configured as a BLE Sensor device, Peripheral role (refer to Section 11 BLE sensor profile demo).

This application uses a new set of APIs allowing to perform the following operations on a BlueNRG-1, BlueNRG-2 Master/Central device:

- Master Configuration Functions
- Master Device Discovery Functions
- Master Device Connection Functions
- Master Discovery Services, Characteristics Functions
- Master Data Exchange Functions
- Master Security Functions
- Master Common Services Functions

These APIs are provided through a binary library and they are fully documented on available doxygen documentation within the DK SW package. The following master/central binary libraries are provided on Bluetooth_LE\Profile_Framework_Central\library folder:

- libmaster_library_bluenrg1.a for IAR, Keil and Atollic toolchains on STSW-BLUENRG1-DK SW package
- master_library_bluenrg1.lib for Keil toolchain on STSW-BNRG_V1-DK SW package
- libmaster_library_bluenrg1.a for IAR and Atollic toolchains on STSW-BNRG_V1-DK SW package

13 BLE HID/HOGP demonstration application

The BLE HID/HOGP demonstration applications are supported by the BlueNRG-1, BlueNRG-2 development platforms (STEVAL-IDB007Vx, STEVAL-IDB008Vx). It demonstrates a BLE device using the standard HID/HOGP Bluetooth low energy application profile. Keyboard and mouse demo examples are provided.

13.1 BLE HID/HOGP mouse demonstration application

The BLE HID mouse application implements a basic HID mouse with two buttons compliant with the standard HID/HOGP BLE application profile.

The HID mouse device is named ‘STMouse’ in the central device list.

The mouse movements are provided by the 3D accelerometer and 3D gyroscope on the BLE development platform.

- The left button is the ‘PUSH1’ button.
- The right button is the ‘PUSH2’ button

If the HID mouse is not used for two minutes, it closes the connection and enters deep sleep mode. This idle connection timeout can be changed from the application. To exit deep sleep mode, press the left ‘PUSH1 button or reset the platform.

13.2 BLE HID/HOGP Keyboard demonstration application

The BLE HID keyboard application implements a basic HID keyboard compliant with the standard HID/HOGP BLE application profile.

The HID mouse device is named ‘STKeyboard’ in the central device list.

To successfully complete the bonding and pairing procedure, insert the PIN: 123456.

To use the HID keyboard:

- Connect the BLE development platform to a PC USB port
- Open a HyperTerminal window (115200, 8, N,1)
- Put the cursor focus on the HyperTerminal window
- The keys that are sent to the central device using the HID/HOGP BLE application profile are also shown on the HyperTerminal window

If the HID keyboard is not used for two minutes, it closes the connection and enters deep sleep mode. This idle connection timeout can be changed from the application. To exit deep sleep mode, press the left ‘PUSH1 button or reset the platform.

14 BLE throughput demonstration application

The BLE throughput demonstration application provides some basic throughput demonstration applications to provide some reference figures regarding the achievable Bluetooth low energy data rate using the BlueNRG-1, BlueNRG-2 device.

The throughput application scenarios provided are:

1. Unidirectional scenario: the server device sends characteristic notifications to a client device.
2. Bidirectional scenario: the server device sends characteristic notifications to a client device and client device sends write without response characteristics to the server device.

The throughput application exposes one service with two (20 byte max.) characteristic values:

- The TX characteristic, with which the client can enable notifications; when the server has data to be sent, it sends notifications with the value of the TX characteristic.
- The RX characteristic, is a writable characteristic; when the client has data to be sent to the server, it writes a value in this characteristic.

The device roles which can be selected are:

1. Server, which exposes the service with the TX, RX characteristics (BLE peripheral device)
2. Client, which uses the service TX, RX characteristics (BLE central device).

Each device role has two instances for each throughput scenario (unidirectional, bidirectional).

The BLE throughput demonstration applications are supported by the BlueNRG-1, BlueNRG-2 development platforms (STEVAL-IDB007Vx, STEVAL-IDB008Vx).

14.1 BLE unidirectional throughput scenario

The unidirectional throughput scenario lets you perform a unidirectional throughput test where a server device sends notification to a client device.

To run this scenario:

- Program the client unidirectional application on one BLE platform and reset it. The platform is seen on the PC as a virtual COM port.
- Open the port in a serial terminal emulator (the required serial port baudrate is 921600)
- Program the server unidirectional application on a second BLE platform and reset it.
- The two platforms try to establish a connection; if successful, the slave continuously sends notifications of TX characteristic (20 bytes) to the client.
- After every 500 packets, the measured application unidirectional throughput is displayed.

14.2 BLE bidirectional throughput scenario

The bidirectional throughput scenario lets you perform a bidirectional throughput test where the server device sends notifications to a client device and client device sends write without response characteristics to the server device.

To run this scenario:

- Program the client bidirectional application on one BLE platform and reset it. The platform is seen on the PC as a virtual COM port.
- Open the related port in a serial terminal emulator (the required serial port baudrate is 921600)
- Program the server bidirectional application on a second BLE platform and reset it.
- Open the related port in a serial terminal emulator (the required serial port baudrate is 921600)

- The two platforms try to establish a connection; if successful, the slave device continuously sends notifications of TX characteristic (20 bytes) to the client device and the client device continuously sends write without responses of the RX characteristic (20 bytes) to the server device.
- After every 500 packets, the measured application bidirectional throughput is displayed.

15 BLE notification consumer demonstration application

The BLE ANCS demonstration application configures a BlueNRG-1, BlueNRG-2 device as a BLE notification consumer, which facilitates Bluetooth accessory access to the many notifications generated on a notification provider.

After reset, the demo places the BLE device in advertising with device name "ANCSdemo" and sets the BlueNRG-1 authentication requirements to enable bonding.

When the device is connected and bonded with a notification provider, the demo configures the BLE notification consumer device to discover the service and the characteristics of the notification provider. When the setup phase is complete, the BLE device is configured as a notification consumer able to receive the notifications sent from the notification provider.

The BLE notification consumer demonstration application is supported by the BlueNRG-1, BlueNRG-2 development platforms (STEVAL-IDB007Vx, STEVAL-IDB008Vx).

16 BLE security demonstration applications

The BLE Security demonstration applications are supported by the BlueNRG-1, BlueNRG-2 development platforms (STEVAL-IDB007Vx, STEVAL-IDB008Vx). They provide some basic examples about how to configure, respectively, two BLE devices as a Central and Peripheral, and setup a secure connection by performing a BLE pairing procedure. Once paired the two devices are also bonded.

The following pairing key generation methods are showed:

- PassKey entry with random pin
- PassKey entry with fixed pin
- Just works
- Numeric Comparison (new paring method supported only from BlueNRG-1, BlueNRG-2 BLE stack v2.x)

For each pairing key generation method, a specific project security configuration is provided for both Central & Peripheral device as shown in the following Table 8. BLE security demonstration applications security configurations combinations. Each Central and Peripheral device must be loaded, respectively, with the application image targeting the proper security configuration, in order to correctly demonstrate the associated BLE security pairing functionality.

Table 8. BLE security demonstration applications security configurations combinations

Pairing key generation method	Central device security configuration	Peripheral device security configuration
PassKey entry with random pin	Master_PassKey_Random	Slave_PassKey_Random
PassKey entry with fixed pin	Master_PassKey_Fixed	Slave_PassKey_Fixed
Just works	Master_JustWorks	Slave_JustWorks
Numeric Comparison	Master_NumericComp	Slave_NumericComp

16.1 Peripheral device

On reset, after initialization, Peripheral device sets security IO capability and authentication requirements, in order to address the selected pairing key generation method, in combinations with the related security settings of the Central device.

After initialization phase, Peripheral device also defines a custom service with 2 proprietary characteristics (UUID 128 bits):

- TX characteristic: notification `(CHAR_PROP_NOTIFY),`

- RX characteristic with properties: read `(CHAR_PROP_READ, GATT_NOTIFY_READ_REQ_AND_WAIT_FOR_APPL_RES` (application is notified when a read request of any type is received for this attribute).

Based on the selected security configuration, the RX characteristic is defined with proper security permission (link must be "encrypted to read" on JustWorks method, link must be "encrypted to read and need authentication to read" on all other methods).

The Peripheral device enters in discovery mode with local name `SlaveSec_Ax` (x= 0,1,2,3 depending on the selected security configuration).

Table 9. Peripheral device advertising local name parameter value

Peripheral device configuration	Advertising local name	Pairing method
Slave_JustWorks	SlaveSec_A0	Just works
Slave_PassKey_Fixed	SlaveSec_A1	PassKey entry with fixed pin

Peripheral device configuration	Advertising local name	Pairing method
Slave_PassKey_Random	SlaveSec_A2	PassKey entry with random pin
Slave_NumericComp	SlaveSec_A3	Numeric Comparison

When a Central device starts the discovery procedure and detects the Peripheral device, the two devices connects.

After connection, Peripheral device starts a slave security request to the Central device `aci_gap_slave_security_req()` and , as consequence, Central devices starts pairing procedure.

Based on the pairing key generation method, user could be asked to perform some actions (i.e. confirm the numeric value if the numeric comparison configuration is selected, add the key, displayed on Peripheral device, on Central hyper terminal, if the passkey entry with random pin configuration is selected).

After devices pairs and get bonded, Peripheral device displays the list of its bonded devices and configures its white list in order to add the bonded Central device to its white list `aci_gap_configure_whitelist()` API.

Central devices starts the service discovery procedure to identify the Peripheral service and characteristics and, then, enabling the TX characteristic notification.

Peripheral device starts TX characteristic notification to the Central device at periodic interval, and it provides the RX characteristic value to the Central device each time it reads it.

When connected, if user presses the BLE platform button PUSH1, Peripheral device disconnects and enters undirected connectable mode mode with advertising filter enabled (`WHITE_LIST_FOR_ALL:` Process scan and connection requests only from devices in the white list). This implies that Peripheral device accepts connection requests only from devices on its white list: Central device is still be able to connect to the Peripheral device; any other device connection requests are not accepted from the Peripheral device.

TX and RX characteristics length is 20 bytes and related values are defined as follow: - TX characteristic value: `{'S','L','A','V','E','_','S','E','C','U','R','I','T','Y','_','T','X',' ',x1,x2};` where x1, x2 are counter values - RX characteristic value: `{'S','L','A','V','E','_','S','E','C','U','R','I','T','Y','_','R','X',' ',x1,x2};` where x1, x2 are counter values

16.2 Central device

On reset, after initialization, Central device uses the `Master_SecuritySet()` API for setting the security IO capability and authentication requirements in order to address the specific selected paring method, in combinations with the related security settings of the Central device. Central device application is using the Central/Master library APIs and callbacks for performing the Central device BLE operations (device discovery, connection, …).

Central device starts a device discovery procedure `(Master_DeviceDiscovery()` API, looking for the associated Peripheral device `SlaveSec_Ax` (x= 0,1,2,3 : refer to Table 9. Peripheral device advertising local name parameter value).

When found, Central connects to the Peripheral device. In order to start the pairing, Central device is expecting the Peripheral device to send a slave security request. Once the security request is received, Central device starts the pairing procedure. Based on the pairing key generation method, user could be asked to perform some actions (i.e. confirm the numeric value if the numeric comparison configuration is selected, add the key, displayed on Peripheral device, on Central hyper terminal, if the passkey entry with random pin configuration is selected). Once the pairing and bonding procedure has been completed, the Central device starts the service discovery procedure in order to find the Peripheral TX & RX characteristics.

After Service Discovery, Central enables the TX characteristic notification. Then the Central device receives periodically the TX characteristic notification value from Peripheral device and read the related RX characteristic value from Peripheral device.

When connected, if user presses the BLE platform PUSH1 button, the Central device disconnects and reconnect to the Peripheral device which enters in undirected connectable mode with advertising filter enabled. Once connected to the Peripheral device, it enters again on the TX characteristic notification/RX characteristic read cycle.

17 BLE power consumption demo application

The BLE power consumption demo application allows to put the selected BLE device in discovery mode: user can select from a test menu which advertising interval to use (100 ms or 1000 ms). To take the BlueNRG-1, BlueNRG-2 current consumption is necessary to connect a DC power analyzer to the JP4 connector of the STEVAL-IDB007Vx, STEVAL-IDB008Vx kit platforms. Further, user can setup a connection with another device configured as a master and take the related power consumption measurements.

The master role can be covered from another BlueNRG-1, BlueNRG-2 kit platform configured with the DTM FW application (DTM_UART_16MHz.hex, DTM_UART_32MHz.hex respectively), and running a specific script through BlueNRG GUI or Script launcher PC applications.

On BLE_Power_Consumption demo application project folder, two scripts are provided for configuring the master device and create a connection with the BlueNRG-1,2 kit platform under test.

The two scripts allow to establish a connection with, respectively, 100 ms and 1000 ms as connection interval.

The power consumption demo supports some test commands:

- f: Device in Discoverable mode with fast interval 100 ms.

- s: Device in discoverable mode with slow interval 1000 ms.

- r: Reset the BlueNRG-1.

- ?: Display this help menu.

Note: This demo application is available only on BlueNRG-1_2 DK SW package (STSW-BLUENRG1-DK) supporting BLE stack v2.x family.

18 BlueNRG-1, BlueNRG-2 peripheral driver examples

The BlueNRG-1, BlueNRG-2 peripheral driver examples applications are supported respectively by the BlueNRG-1, BlueNRG-2 development platforms (STEVAL-IDB007Vx, STEVAL-IDB008Vx). The kit contains a set of examples demonstrating how to use the BlueNRG-1, BlueNRG-2 device peripheral drivers ADC, GPIOs, I²C, RTC, SPI, Timers, UART and WDG.

Note: On all following sub-sections, any reference to the BlueNRG-1 device and related kit platform STEVAL-IDB007Vx (with x=1, 2) is also valid for the BlueNRG-2 device and related kit platform STEVAL-IDB008Vx (with x=1, 2).

18.1 ADC examples

ADC polling: conversion is managed through the polling of the status register. The systick timer is used to have a delay of 100 ms between two samples. Each sample from ADC is printed through UART (USB-to-SERIAL must be connected to the PC). The default input is the differential ADC1-ADC2.

ADC DMA: conversion is managed through the ADC DMA channel. The systick timer is used to have a delay of 100 ms between two samples. Each sample from ADC is printed through UART (USB-to-SERIAL must be connected to the PC).

ADC PDM: this example shows a PDM stream processor from a MEMS microphone (MP34DT01-M i.e.) to UART. The application also supports the MEMS microphone MP34DT01-M available on X-NUCLEO-CCA02M1 board (refer to related BlueNRG-1 DK SW package ADC PDM doxygen documentation for HW connections setup).

User is requested to connect the BLE platform to a PC USB port and open PuTTY serial terminal [512000, 8-N-1-N]. PuTTY serial terminal has to be configured for storing the captured data on a log file. After the data have been captured, the PC Audacity tool can been opened for importing the streamed data, following these steps:

- File/Import/Raw Data.
- Open the log data.
- Configure as follows:
  - Encoding: Signed 16-bit PCM.
  - Byte order: Little-endian.
  - Channels: 1 Channel (Mono).
  - Sample rate: 8000 (default, 16 kbps is supported by changing the firmware symbol FS in ADC_PDM_main.c)
  - Press the button Import.
- Play the audio.

Note: Due to the fact that the output data format is 2-bytes (B1B2), it is possible that the serial terminal gets as first byte, half data (B2). This first byte must be removed from the log file.

18.2 Flash example

Data storage: demonstrates basic flash operations as erase, write and verification.

18.3 GPIO examples

Input interrupt: demonstrates the use of GPIO input interrupts.

- The PUSH1 button (IO13) is configured to generate the interrupt event on both edges of the input signal. LED DL1 is toggled ON if the level is high and OFF if low.
- The PUSH2 button (IO5) is configured to generate the interrupt event on the rising edge of the input signal. LED DL2 is toggled ON/OFF at each rising edge event.

IO toggle: demonstrates GPIO state changes by toggling LEDs DL1 and DL2 every 500 ms.

IO wakeup: demonstrates device wakeup from standby mode using the GPIO interrupt.

- The PUSH1 button (IO13) is configured to generate the interrupt event on both edges of the input signal. LED DL2 is toggled, the system becomes active and LED DL1 is toggled by the systick interrupt service routine every 500 ms.

Once the device is in standby, there is no way to open a connection with the debug tool or download new code as the clocks are down and the system voltages are at their minimum values. It is necessary to wake-up the system first, which is why the IO9 (SDW clock signal) is wake-up event. In this case, any connection attempt from the debugger wakes up the system.

18.4 I²C examples

In all of the following examples, I²C is configured in master mode and its clock frequency is set to 10 kHz.

Master polling: I²C communication is controlled by polling the I²C status register content. This example involves a master board with Master_Polling firmware code and a slave board with Slave_Polling firmware.

The Master board has a small command line interface through UART (USB-to-SERIAL must be connected to the PC), which you can use to read and change the LED status of the slave board. I²C is used to transfer the information and change the status of the LEDs on the slave board.

Slave polling: I²C communication is controlled by polling the I²C status register content. This also involves a master and a slave board with respective Master_Polling and Slave_Polling firmware. The slave board receives read and change requests for the LEDs via I²C.

Master sensor: I²C communication is controlled by polling of I²C status register content, interrupts or DMA (three different configurations). In this example, the environmental sensor LPS25HB is configured to provide output data at 1 Hz. The BlueNRG-1 polls the status register of the sensor and prints available pressure and temperature data via UART (USB-to-SERIAL must be connected to the PC).

18.5 Micro examples

Hello world: example for the basic ‘BlueNRG-1 Hello World’ application. Connect the BlueNRG-1 platform to a PC USB port and open a specific PC tool/program (like Tera Term): the "Hello World: BlueNRG-1 is here!" message is displayed.

Sleep test: this test provides an example for the following BlueNRG-1 sleep modes:

- SLEEPMODE_WAKETIMER places the BlueNRG-1 in deep sleep with the timer clock sources running. The wakeup sources are typing any character on the keyboard, the PUSH1 button or the sleep timer configured with a timeout of 5 s.
- The SLEEPMODE_NOTIMER places the BlurNRG-1 in deep sleep with the sleep timer clock sources turned off. The only wakeup sources are typing any character on the keyboard and the PUSH1 button.

The demo supports some user commands:

- **s**: SLEEPMODE_NOTIMER - wake on UART/PUSH1
- **t**: SLEEPMODE_WAKETIMER - wake on UART/timeout 5 s/PUSH1
- **l**: Toggle led DL1
- **p**: Print the ‘Hello World’ message
- **r**: Reset the BlueNRG-1 device
- **?**: Display this help menu
- **PUSH1**: toggle LED DL1

18.6 Public Key Accelerator (PKA) demonstration application

The BlueNRG-1 PKA demonstration application is supported by the BlueNRG-1, BlueNRG-2 development platforms. It provides a basic examples about how to use the available PKA driver APIs in order to perform a basic PKA processing and check the results.

The Public Key Accelerator (PKA) is a dedicated HW block used for computation of cryptographic public key primitives related to ECC (Elliptic curve Cryptography).

Please notice that this peripheral is used by the BlueNRG-1, BlueNRG-2 Bluetooth Low Energy stack during the security pairing procedures, so user application must not use it during such procedures.

The PKA demonstration applications performs the following steps:

Starting from the PKA known point on the ellipse `PKS_SetData()` with `PKA_DATA_PCX, PKA_DATA_PCY`and from a random generated keyA, it performs a PKA process which generates a new point A on the ellipse.

The same process is repeated from a new generated random keyB, leading to a new point B on the ellipse.

At this stage, a new PKA process is started using the keyA with the point B coordinates. This generates a new point C which is still on the same ellipse.

18.7 RNG examples

Terminal: shows how to use the RNG. It gets the RNG values and print them on the terminal.

18.8 RTC examples

Clock watch: implements both RTC timer and RTC clockwatch.

The RTC timer generates the 500 ms interrupt interval. The LED DL1 state is toggled in the RTC interrupt handler to signal proper RTC timer operation.

The RTC clockwatch is also enabled with the system time and date set to December 1st 2014, 23 h 59 m 31 s. The RTC clockwatch match registers are then set to December 2nd 2014, 0 h 0 m 1 s. As soon as the RTC clockwatch data register and match registers coincide (30 s after device power up), the RTC clockwatch match interrupt is generated and LED DL2 is toggled to signal the event.

Time base: the RTC is configured in the periodic timer mode, the load register (RTC_TLR1) value is set and the RTC is enabled. Whenever the RTC timer reaches the value 0x00 it generates an interrupt event and the timer value is automatically re-loaded from the RTC_TLR1 register, which is set to generate the interrupt every 1 s. The LED DL1 is toggled at each interrupt event.

Time base pattern: periodic mode is used with a pattern configuration. The RTC is configured in the periodic timer mode and register RTC_TLR1 is set to generate a 1 s interval, while RTC_TLR2 is set to generate a 100 ms interval.

The RTC is then enabled and whenever the RTC timer reaches the value 0x00 it generates an interrupt and the timer value is automatically re-loaded from register RTC_TLR1 or RTC_TLR2 register depending on the pattern register setting.

The pattern size is set to 8 bits and the pattern is set to 0b11110010, so the RTC generates four intervals with the RTC_TLR1 value followed by two RTC_TLR2 value intervals. The pattern repeats itself and the RTC interrupt routine toggles LED DL1 (IO6).

18.9 SPI examples

The following SPI application examples are available:

Master polling: involves a master board with the Master_Polling firmware code and a slave board with the Slave_Polling firmware. The Master board has a small command line interface through UART (USB-to-SERIAL must be connected to the PC), which you can use to read and change the LED status of the slave board via SPI.

The SPI is configured in master mode and the SPI clock set to 100 kHz. The data is transferred in the Motorola format with an 8-bit data frame, with clock low when inactive (CPOL=0) and data valid on clock trailing edge (CPHA = 1).

Slave polling: SPI communication is controlled by polling the SPI status register content. This also involves a master and a slave board with respective Master_Polling and Slave_Polling firmware. The slave board receives read and change requests for the LEDs via SPI.

The SPI is configured in slave mode and the SPI clock set to 100 kHz. The data is transferred in the Motorola format with an 8-bit data frame, with clock low when inactive (CPOL=0) and data valid on clock trailing edge (CPHA = 1).

Master sensor: SPI communication is controlled by polling of the SPI status register content, interrupts or DMA (3 different configurations). SPI is used to communicate with the LSM6DS3 inertial sensor SPI interface. Whenever the sensor generates an IRQ, the accelerometer and gyroscope output data are read and printed through UART (USB-to-SERIAL must be connected to the PC).

The SPI is configured in master mode and the SPI clock set to 100 kHz. The data is transferred in the Motorola format with an 8-bit data frame, with clock low when inactive (CPOL=0) and data valid on clock trailing edge (CPHA = 1).

Master DMA: SPI communication is controlled by DMA of the SPI status register content. It involves a master board with the Master_Dma firmware code and a slave board with the Slave_Dma firmware. The Master board has a small command line interface through UART (USB-to-SERIAL must be connected to the PC), which you can use to read and change the LED status of the slave board via SPI.

The SPI is configured in master mode and the SPI clock set to 100 kHz. The data is transferred in the Motorola format with an 8-bit data frame, with clock low when inactive (CPOL=0) and data valid on clock trailing edge (CPHA = 1).

Slave DMA: SPI communication is controlled by DMA of the SPI status register content. It involves a master board with the Master_Dma firmware code and a slave board with the Slave_Dma firmware. The slave board receives read and change requests for the LEDs via SPI.

The SPI is configured in slave mode and the SPI clock set to 100 kHz. The data is transferred in the Motorola format with an 8-bit data frame, with clock low when inactive (CPOL=0) and data valid on clock trailing edge (CPHA = 1).

SPI 3 wires: demonstrates the SPI 3 wires communication for reading humidity and temperature data from the humidity sensor HTS221. In this example, the evaluation board for HTS221, STEVAL-MKI141V2 is used. The SPI clock frequency is set to 100 kHz. The data is transferred in the Microwire format and the data frame size is 8 bits.

18.10 SysTick examples

Time base: The interrupt service routine toggles the user LEDs at approximately 0.5 s intervals.

18.11 Timers examples

Mode 1: Timer/Counter 1 (TnCNT1) functions as the time base for the PWM timer and counts down at the clock rate selected by the Timer/Counter 1 clock selector. When an underflow occurs, the timer register is reloaded alternately from the TnCRA (first reload) and TnCRB registers and count down begins from the loaded value.

Timer/Counter 2 can be used as a simple system timer, an external-event counter, or a pulse-accumulate counter. Counter TnCNT2 counts down with the clock selected by the Timer/Counter 2 clock selector, and can be configured to generate an interrupt upon underflow.

MFTX1 and MFTX2 use prescaled clock as Timer/Counter 1. The IO2 pin is configured as output, generating a signal with 250 ms positive level and 500 ms negative level via MFTX1. The IO3 pin is configured as output, generating a signal with 50 ms positive level and 100 ms negative level via MFTX2.

Timer/Counter 1 interrupts upon reload are enabled for MFTX1 and MFTX2; interrupt routines toggle LED DL1 for MFTX1 and LED DL2 for MFTX2.

Mode 1a (pulse-train mode): the Timer/Counter 1 functions as PWM timer and Timer/Counter 2 is used as a pulse counter that defines the number of pulses to be generated.

In this example, MFTX2 is configured to generate 30 pulses with positive level of 500 ms and negative level of 250 ms. MFTX2 uses prescaled clock as Timer/Counter 1. The IO3 pin is configured as output generating the number of pulses configured.

Interrupts TnA and TnB are enabled and toggle GPIO 8 and 10, while Interrupt TnD is enabled and sets GPIO 7.

A software start trigger or external rising or falling edge start trigger can be selected. This example uses a software trigger which is generated after system configuration.

Timer/Counter 1 interrupts on reload are enabled for MFTX1. Interrupt routines toggle LED DL1 for MFTX2.

Mode 2 (dual-input capture mode): Timer/Counter 1 counts down with the selected clock and TnA and TnB pins function as capture inputs. Transitions received on the TnA and TnB pins trigger a transfer of timer content to the TnCRA and TnCRB registers, respectively. Timer/Counter 2 counts down with selected clock and can generate an interrupt on underflow.

In this example, MFTX1 is used. The CPU clock is selected as the clock signal for Timer/Counter 1 and a Prescaled clock is used as the clock source for Timer/Counter 2.

Sensitivity to falling edge is selected for TnA and TnB inputs; counter preset to 0xFFFF is disabled for both inputs.

The IO2 pin is internally connected to TnA input (MFTX1) and the IO3 pin is internally connected to TnB input (MFTX1).

Interrupts TnA and TnB are enabled and triggered by transitions on pins TnA and TnB, respectively. The interrupt routine records the value of TnCRA or TnCRB and calculates the period of the input signal every second interrupt.

Interrupt TnC is enabled and is triggered on each underflow of Timer/Counter1; it increments the underflow counter variables used to calculate the input signal period.

LED DL1 is toggled ON if a frequency of about 1 kHz is detected on IO2, and LED DL2 is toggled ON if a frequency of about 10 kHz is detected on IO3.

Mode 3 (dual independent timer/counter): the timer/counter is configured to operate as a dual independent system timer or dual external-event counter. Timer/Counter 1 can also generate a 50% duty cycle PWM signal on the TnA pin, while the TnB pin can be used as an external-event input or pulse-accumulate input, and serve as the clock source to either Timer/Counter 1 or Timer/Counter 2. Both counters can also be operated from the prescaled system clock.

In this example MFTX1 is used. The CPU clock is selected as the clock signal for Timer/Counter 1, while Timer/Counter 2 uses an external clock on TnB pin. Sensitivity to rising edge is selected for TnB input. Timer/Counter 1 is preset and reloaded to 5000, so the frequency of the output signal is 1 kHz. Timer/Counter 2 is preset and reloaded to 5.

The IO3 pin is internally connected to TnA input (MFTX1), while the IO2 pin is configured as output and configured as the PWM output from Timer/Counter 1.

The LED DL1 is toggled in the main program according to a variable which is changed in TnD interrupt routine. Interrupt TnA and TnD are enabled and are triggered on the underflow of Timer/Counter1 and Timer/Counter2 respectively.

Mode 4 (input-capture plus timer): is a combination of mode 3 and mode 2, and makes it possible to operate Timer/Counter 2 as a single input-capture timer, while Timer/Counter 1 can be used as a system timer as described above.

In this example, MFTX1 is used. The CPU clock is selected as the input clock for Timer/Counter 1 and Timer/Counter 2. Automatic preset is enabled for Timer/Counter 2.

The IO2 pin is internally connected to the TnB input (MFTX1), while the IO3 pin is configured as the output and configured as the PWM output from Timer/Counter 1.

Interrupt TnA is enabled and triggered on the underflow of Timer/Counter1; it sets a new value in the TnCRA register. Interrupt TnB in enabled and triggered when a transition on TnB input (input capture) is detected; it saves the TnCRB value. Interrupt TnD in enabled and it triggered on the underflow of Timer/Counter2.

MFT timers: this example shows how configure peripherals MFT1, MFT2 and SysTick to generate three timer interrupts at different rate: MFT1 at 500 ms, MFT2 at 250 ms and SysTick at 1 second.

18.12 UART examples

DMA: IO8 and IO11 are configured as UART pins and DMA receive and transmit requests are enabled. Each byte received from UART is sent back through UART in an echo application (USB-to-SERIAL must be connected to the PC).

Interrupt: IO8 and IO11 are configured as UART pins and receive and transmit interrupts are enabled. Each byte received from UART is sent back through UART in an echo application (USB-to-SERIAL must be connected to the PC).

Polling: IO8 and IO11 are configured as UART pins. Each byte received from UART is sent back through UART in an echo application (USB-to-SERIAL must be connected to the PC).

18.13 WDG examples

Reset: demonstrates the watchdog functionality and using it to reboot the system when the watchdog interrupt is not serviced during the watchdog period (interrupt status flag is not cleared).

The watchdog is configured to generate the interrupt with a 15 s interval, then it is enabled and monitors the state of the PUSH1 button (IO13 pin). Any change on this pin triggers the watchdog counter to reload and restart the 15 s interval measurement.

If the IO13 pin state does not change during this interval, the watchdog generates an interrupt that is intentionally not cleared and therefore remains pending; the watchdog interrupt service routine is therefore called continuously and the system is stuck in the watchdog interrupt handler.

The chip is reset as it can no longer execute user code. The second watchdog timeout triggers system reboot as a new watchdog interrupt is generated while the previous interrupt is still pending. The application then starts measuring the 15 s interval again.

The three user LEDs are toggled at increasing frequencies until the board is reset or PUSH1 button is pressed, which restores the LEDs toggling frequency with the 15 s watchdog timer.

Wakeup: The watchdog timer is a 32-bit down counter that divides the clock input (32.768 kHz) and produces an interrupt whenever the counter reaches zero. The counter is then reloaded with the content of the WDT_LR register. If the interrupt status flag is not cleared and a new interrupt is generated, then the watchdog may generate a system reset.

This example demonstrates the use of the watchdog to periodically wake the system from standby mode using the watchdog interrupt. The watchdog is configured to generate the interrupt at 1 s intervals. The watchdog is then enabled and the system is switched to the standby mode. As soon as the watchdog interrupt is generated, the system wakes up, LED1 (IO6 pin) is toggled and the device returns to standby mode. The IO6 pin is therefore toggled every 1 s.

19 Schematic diagrams

Figure 27. **STEVAL-IDB007V1 arduino connectors**

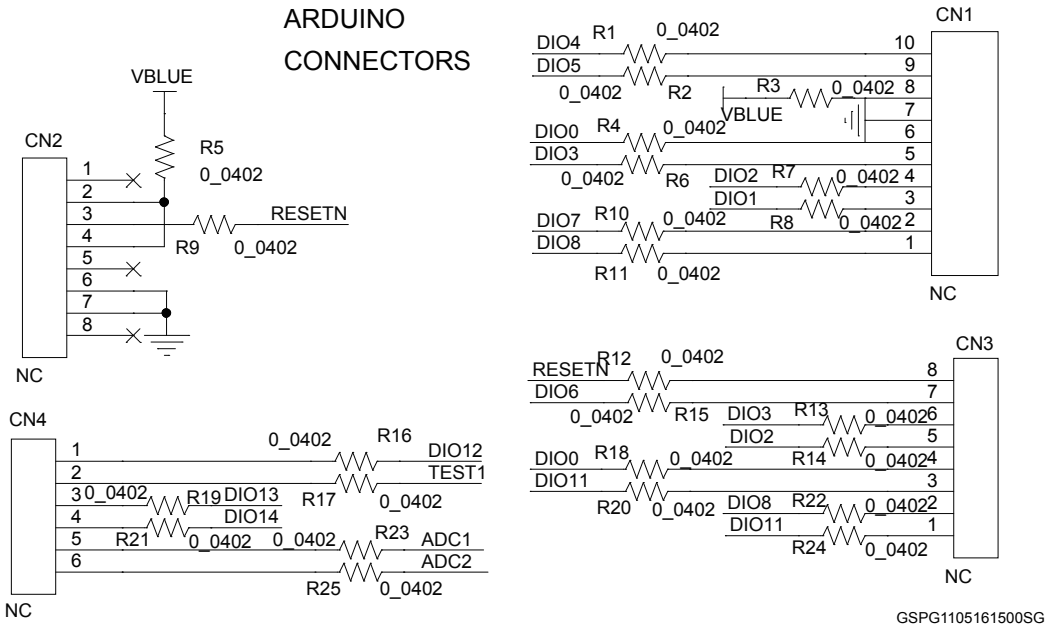


Figure 28. **STEVAL-IDB007V1 JTAG**

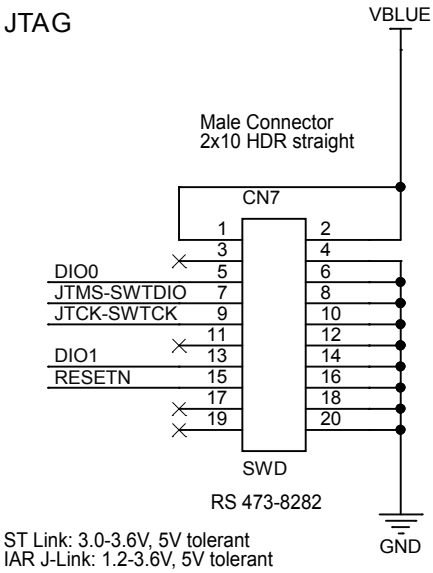


Figure 29. **STEVAL-IDB007V1 BlueNRG-1**

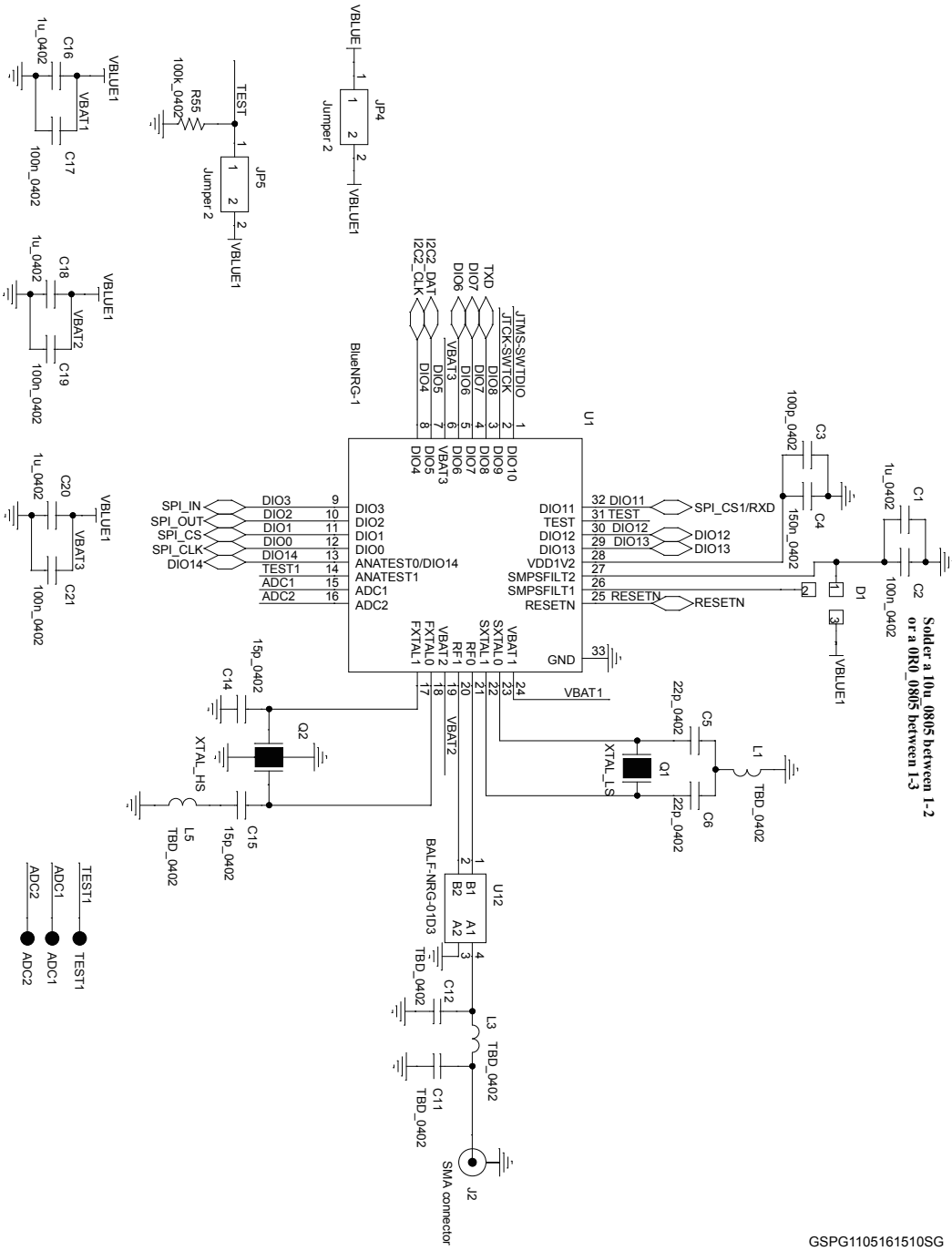


Figure 30. **STEVAL-IDB007V1 power management, sensors**

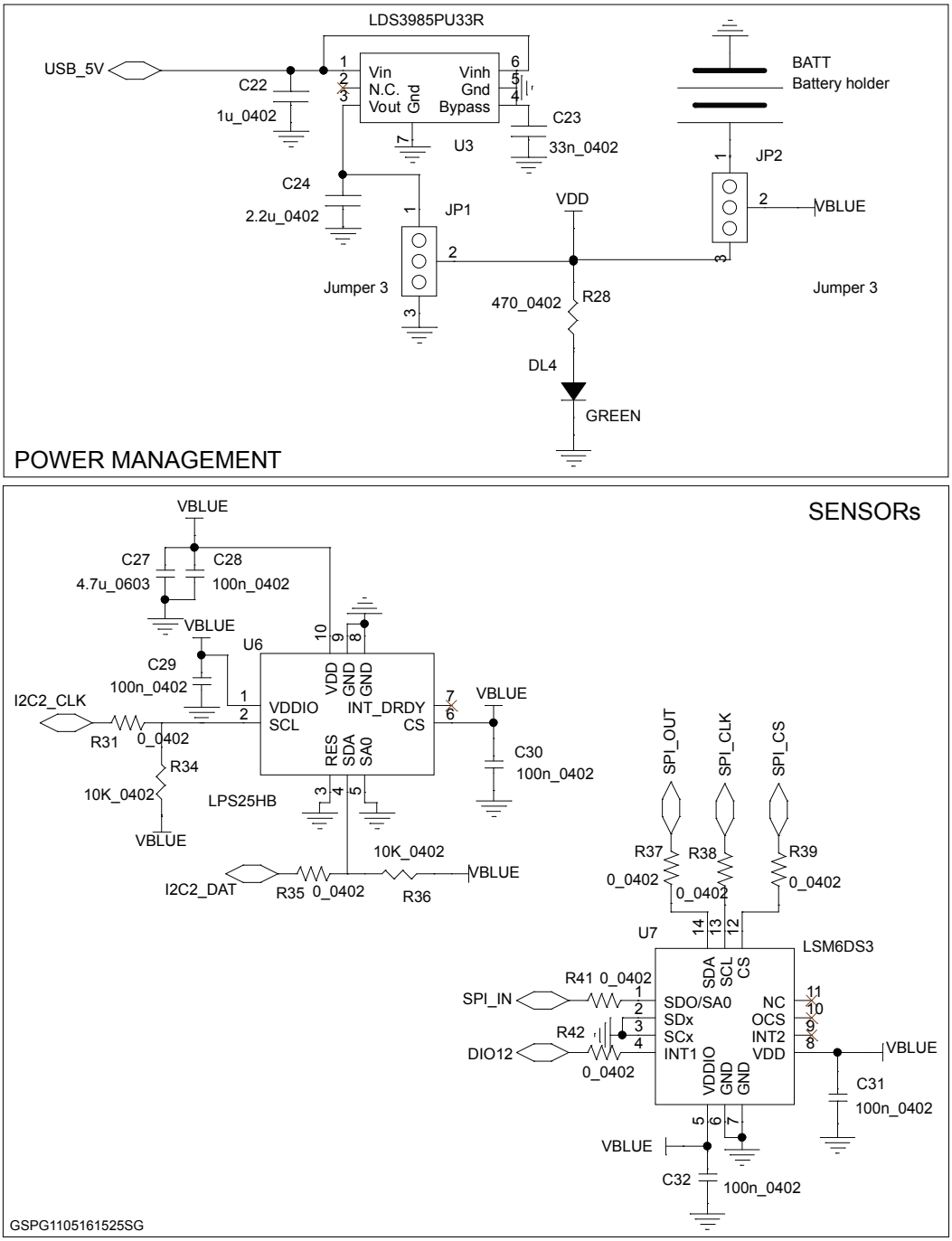


Figure 31. **STEVAL-IDB007V1 buttons and LEDs**

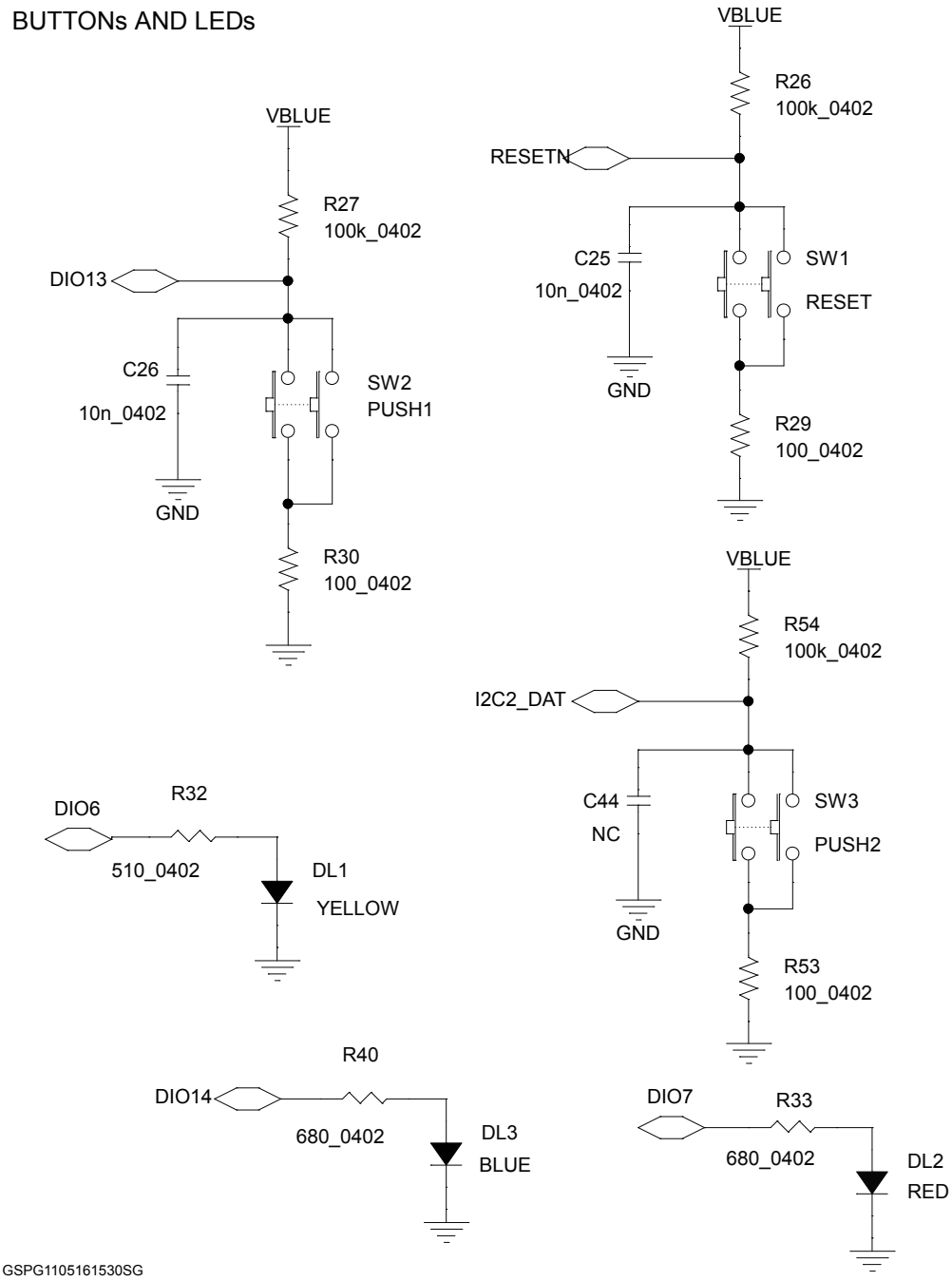


Figure 32. **STEVAL-IDB007V1 micro**

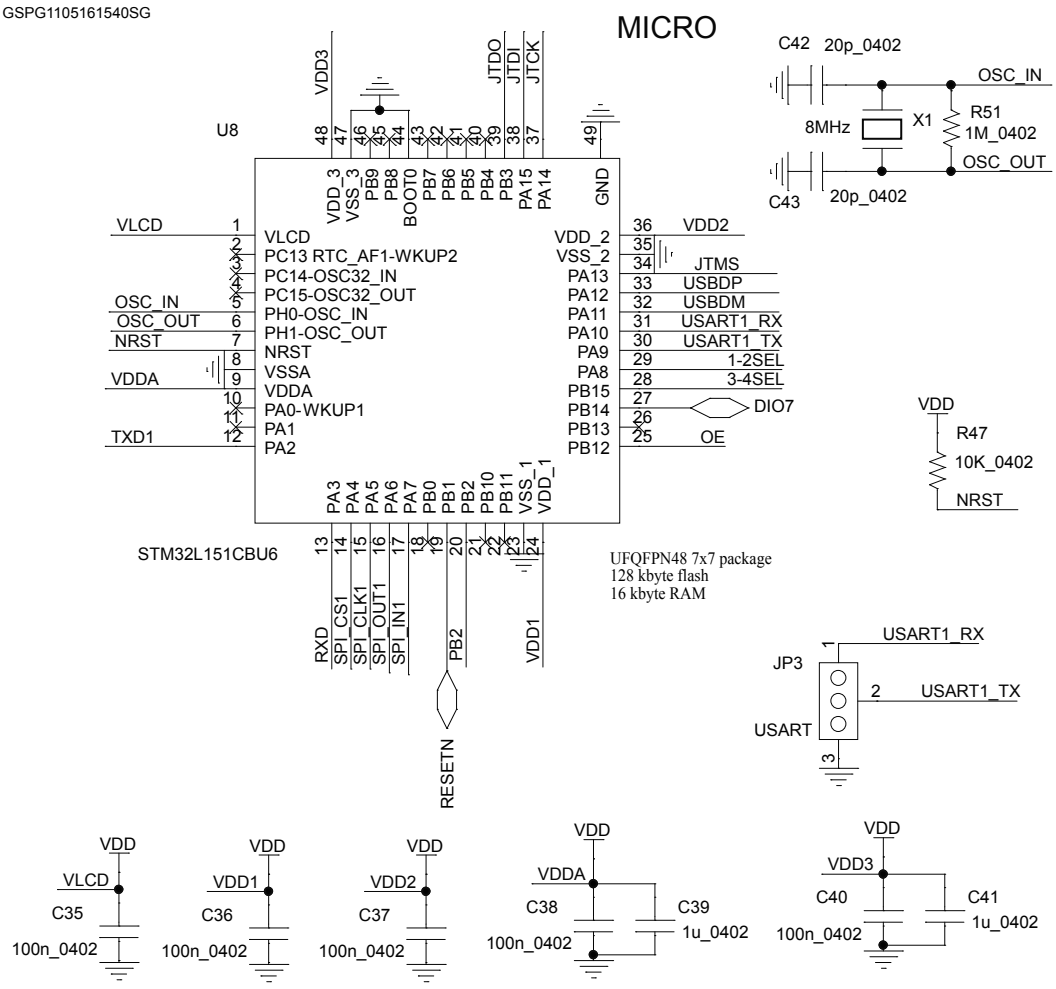


Figure 33. **STEVAL-IDB007V1 USB, level translator, JTAG for micro**

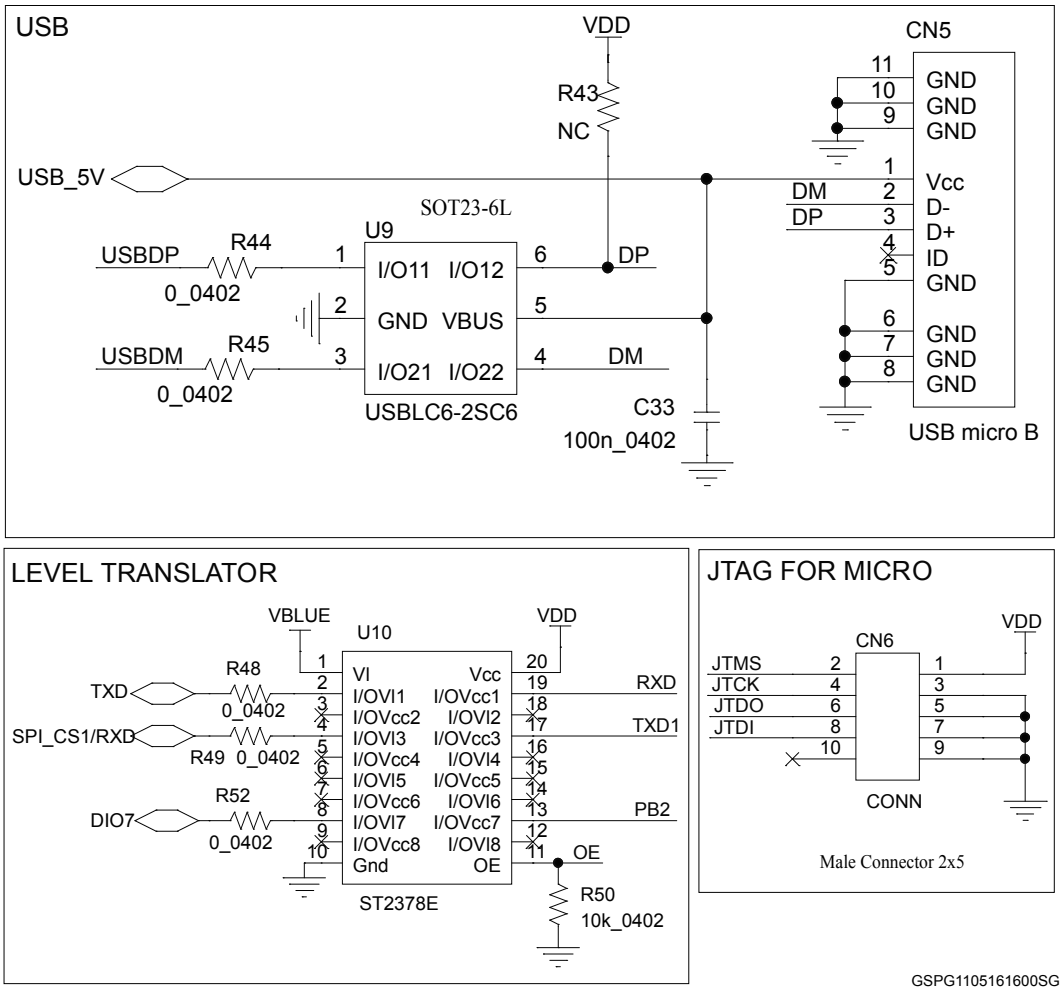


Figure 34. **STEVAL-IDB007V1 switch**

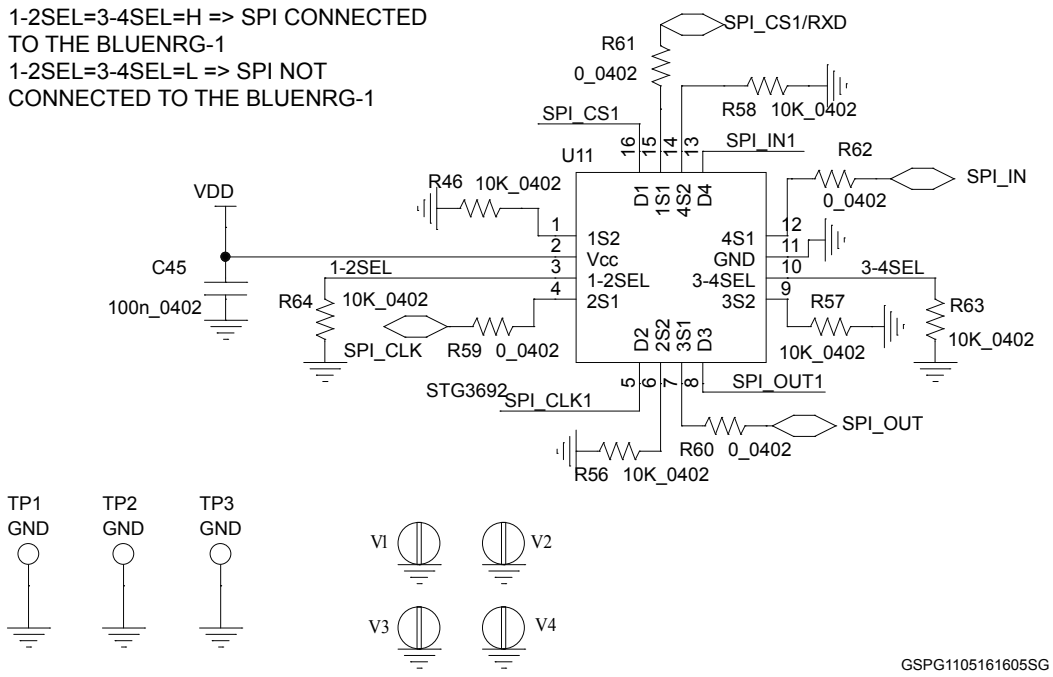


Figure 35. **STEVAL-IDB008V1 circuit schematic JTAG**

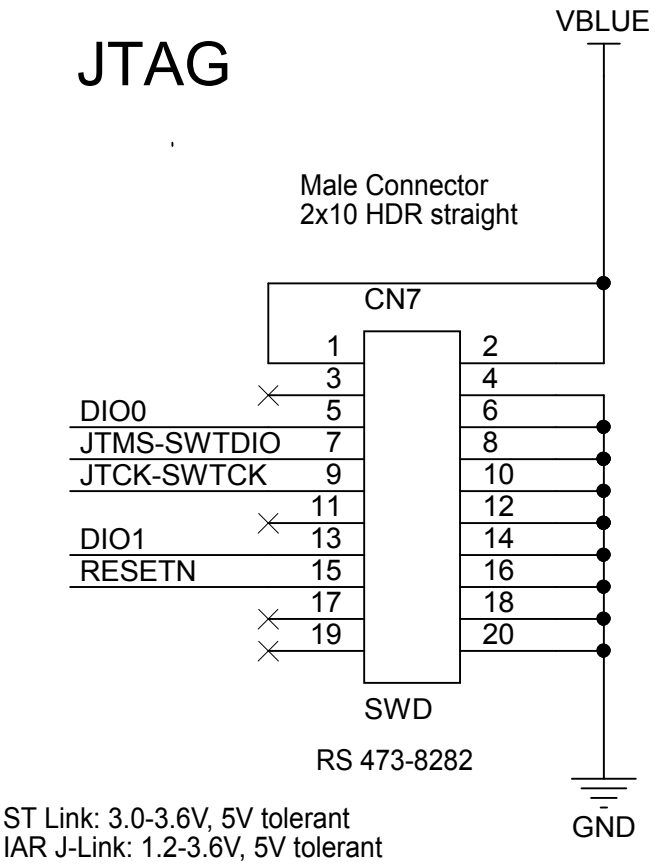


Figure 36. **STEVAL-IDB008V1 circuit schematic arduino connectors**

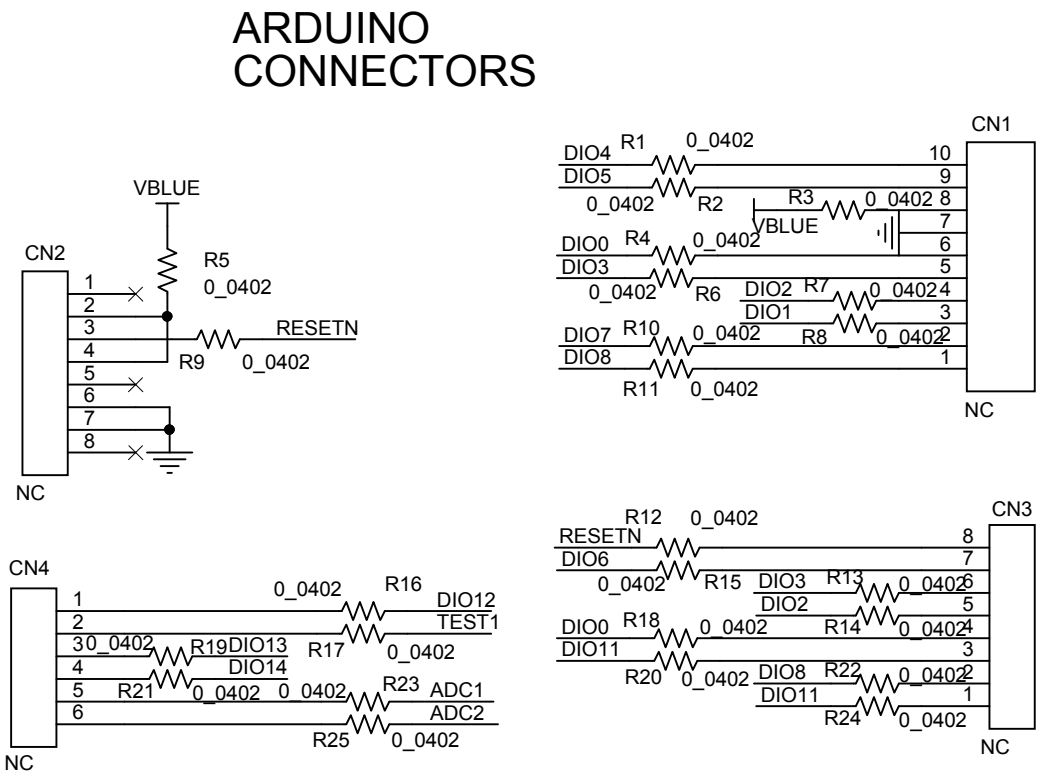


Figure 37. **STEVAL-IDB008V1 BlueNRG-2**

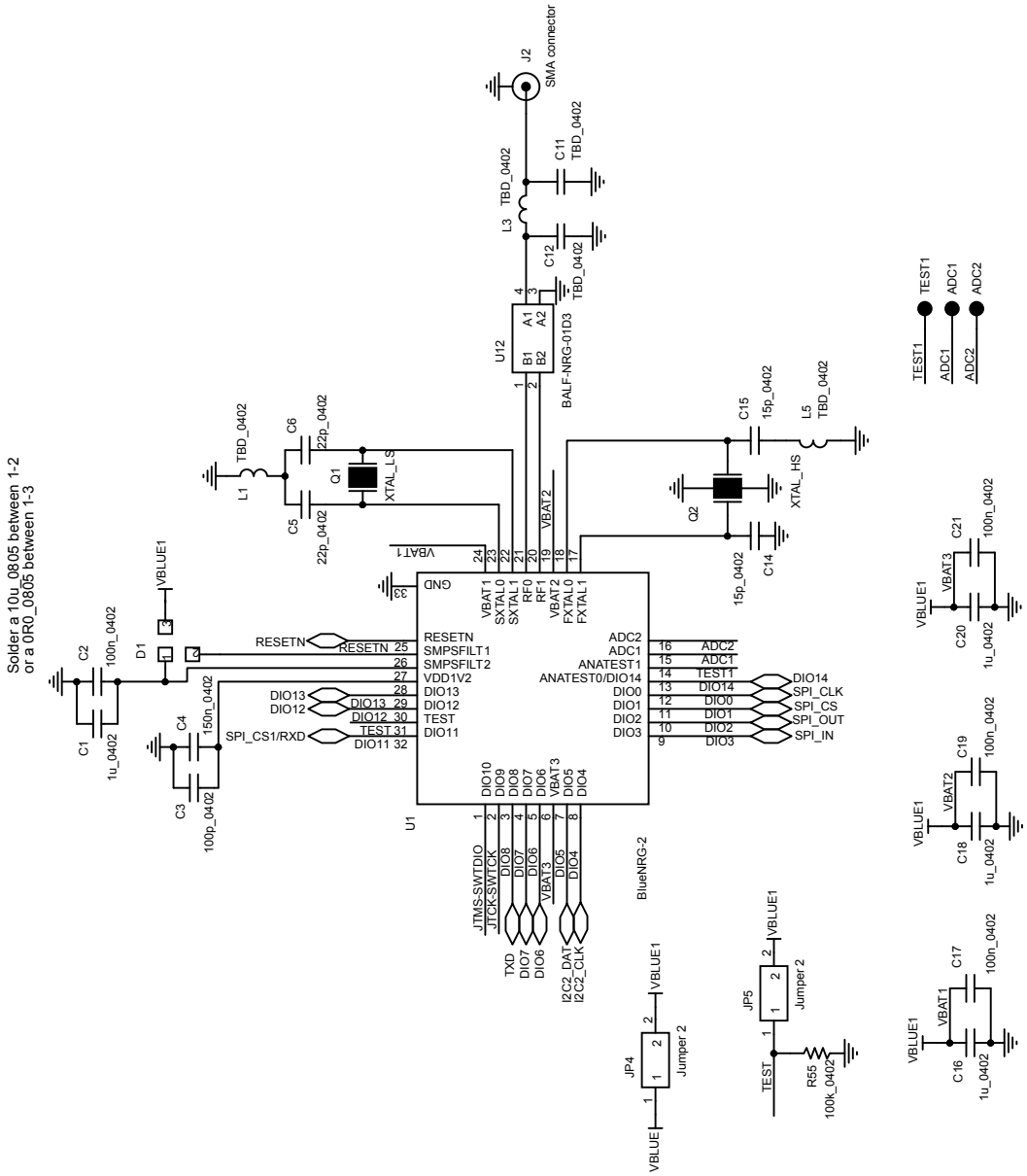


Figure 38. **STEVAL-IDB008V1 buttons and leds**

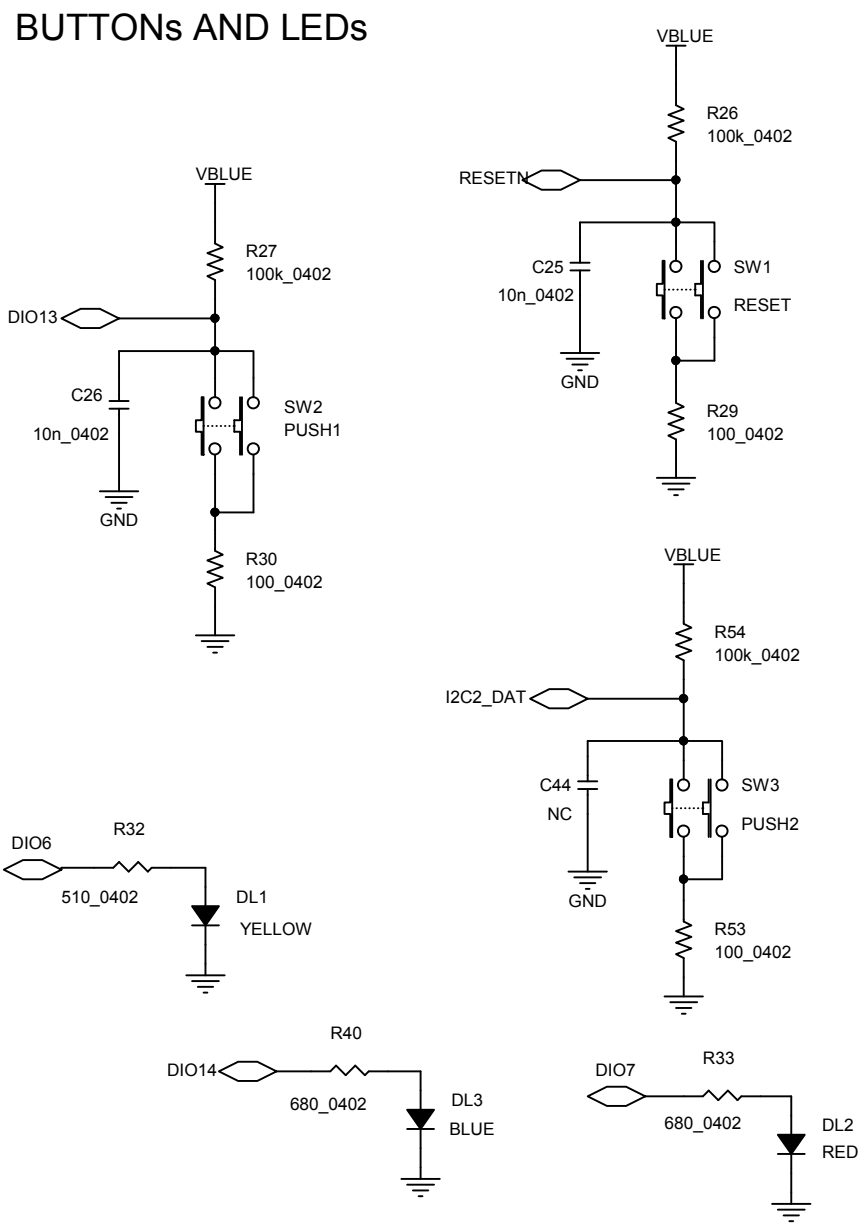


Figure 39. **STEVAL-IDB008V1 sensors**

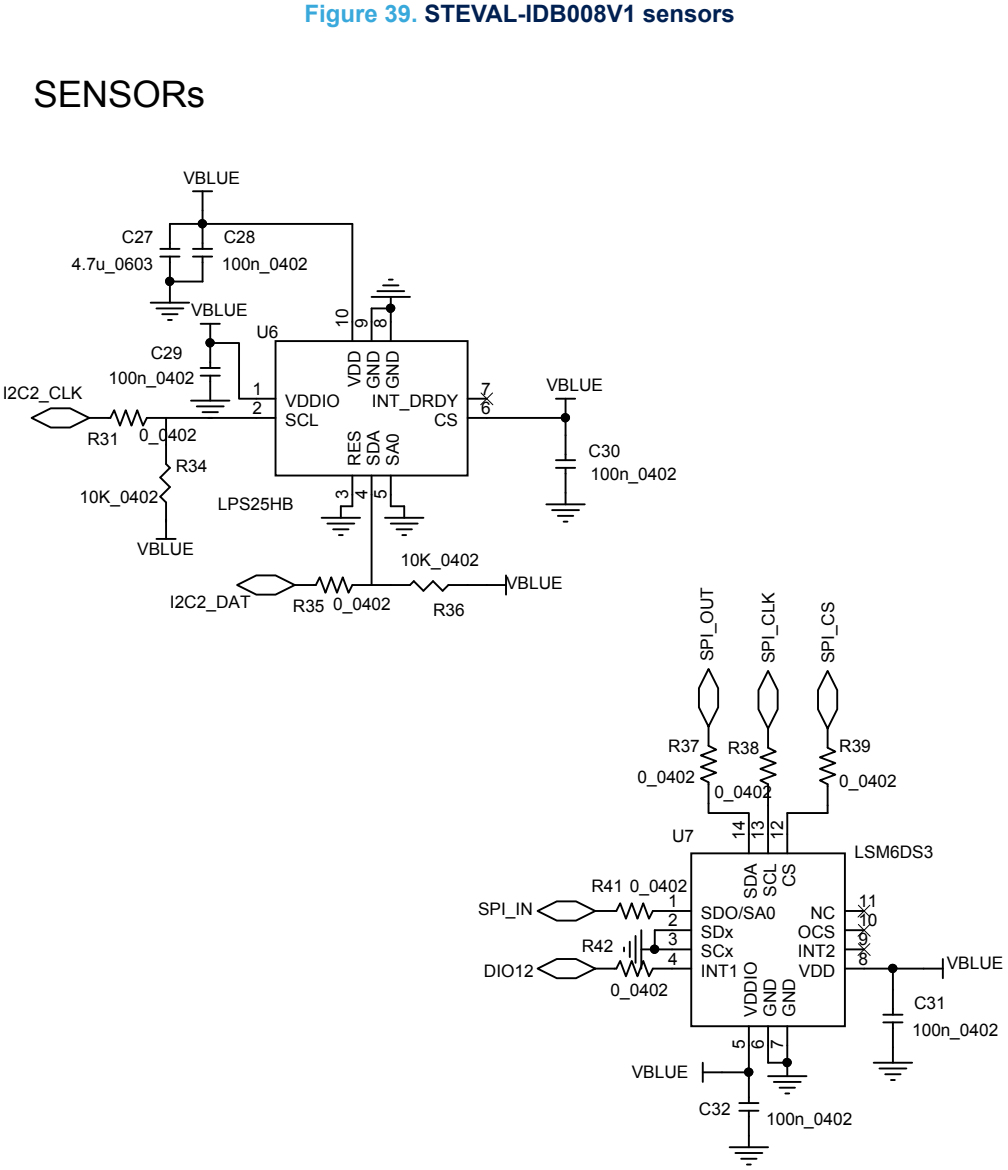


Figure 40. **STEVAL-IDB008V1 power management**

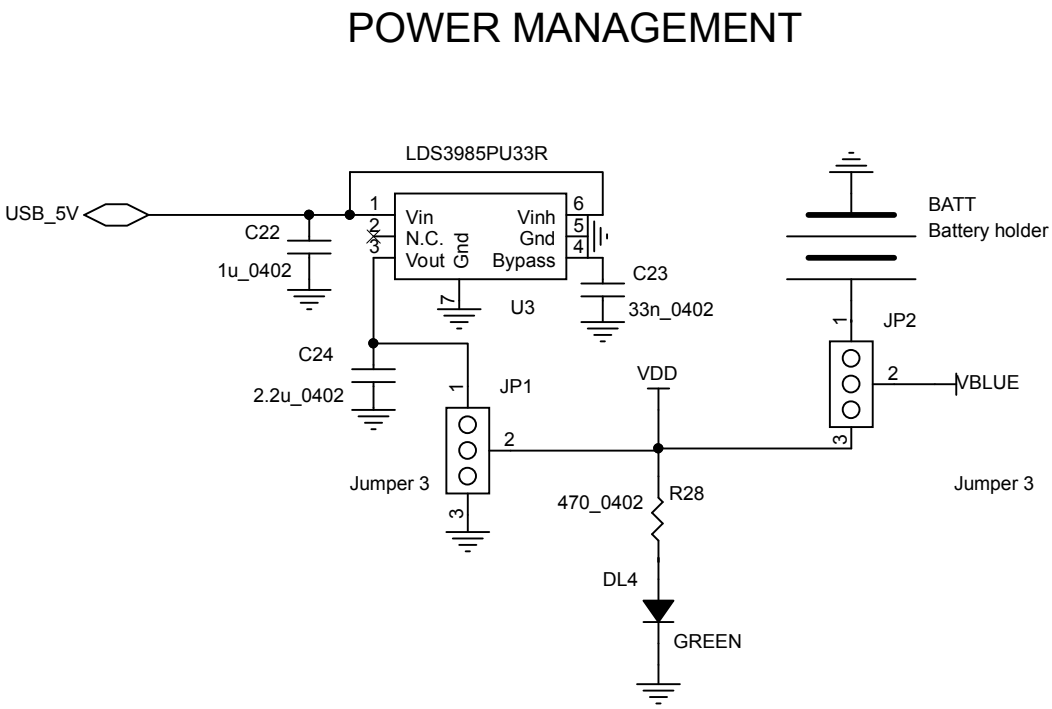
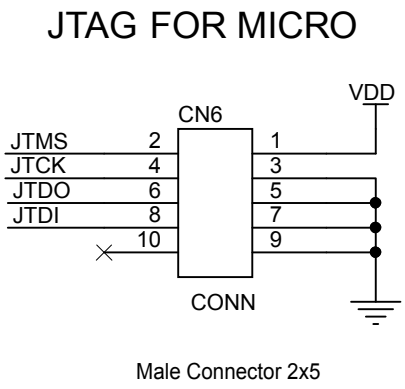


Figure 41. **STEVAL-IDB008V1 JTAG for micro**



Male Connector 2x5

Figure 42. **STEVAL-IDB008V1 USB**

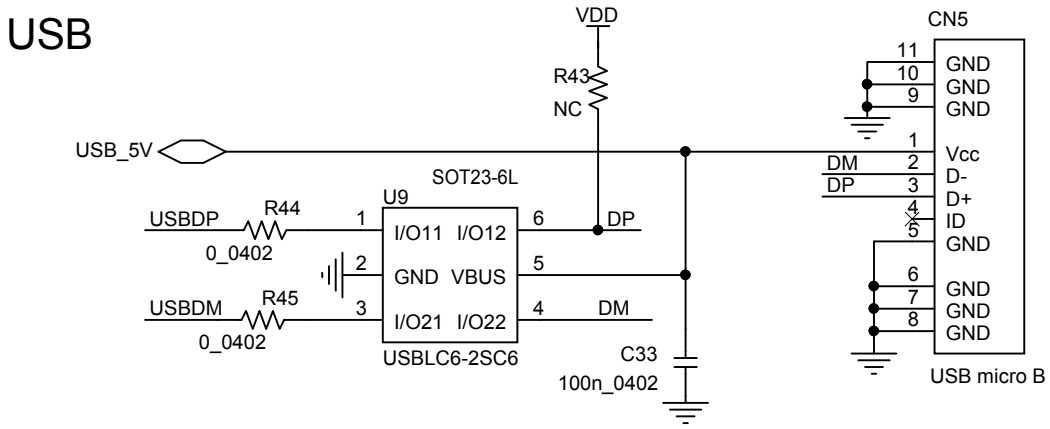


Figure 43. **STEVAL-IDB008V1 circuit schematic TP1, TP2, TP3**

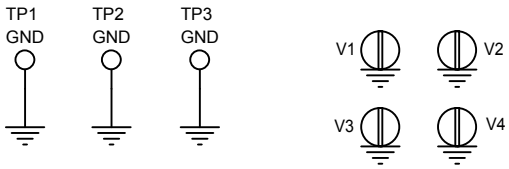


Figure 44. **STEVAL-IDB008V1 switch**

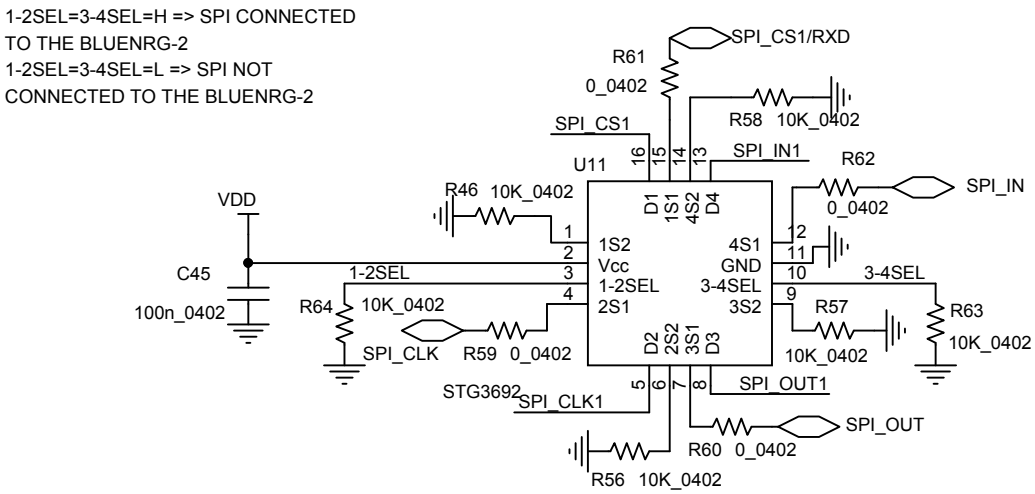


Figure 45. **STEVAL-IDB008V1 micro**

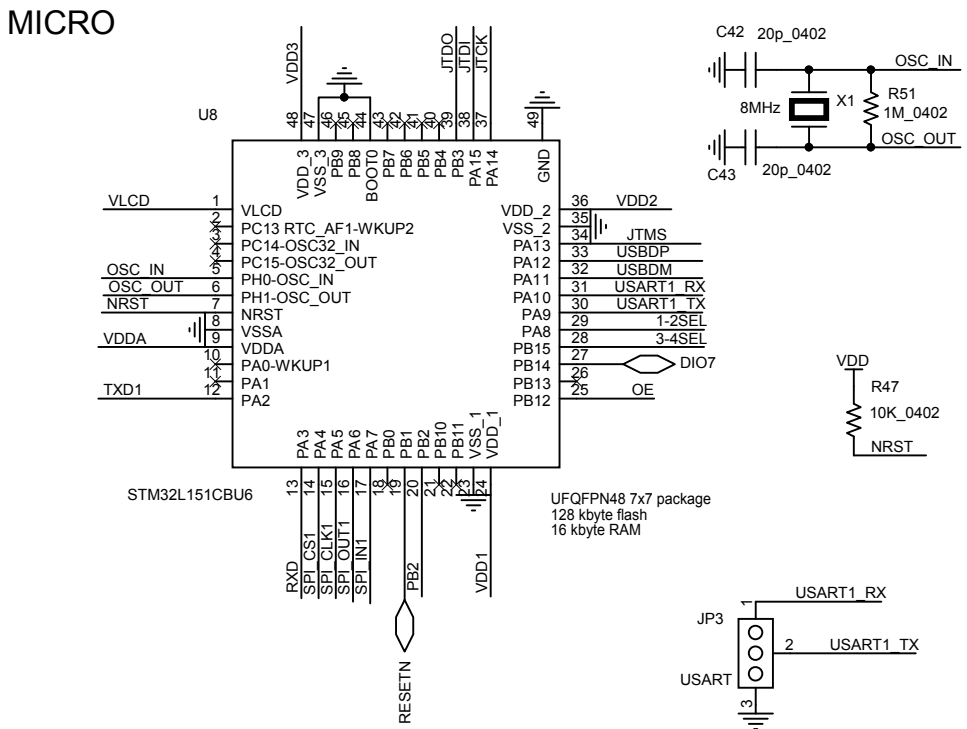
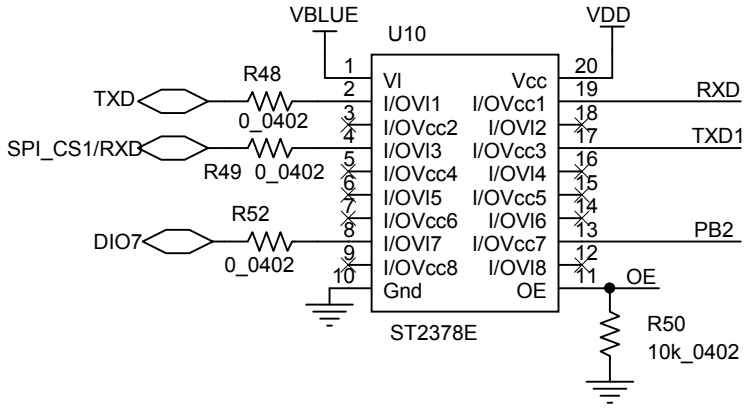


Figure 46. **STEVAL-IDB008V1 level translator**



20 Schematic diagrams for STEVAL-IDB007V2

Figure 47. **STEVAL-IDB007V2 - scheme 1**

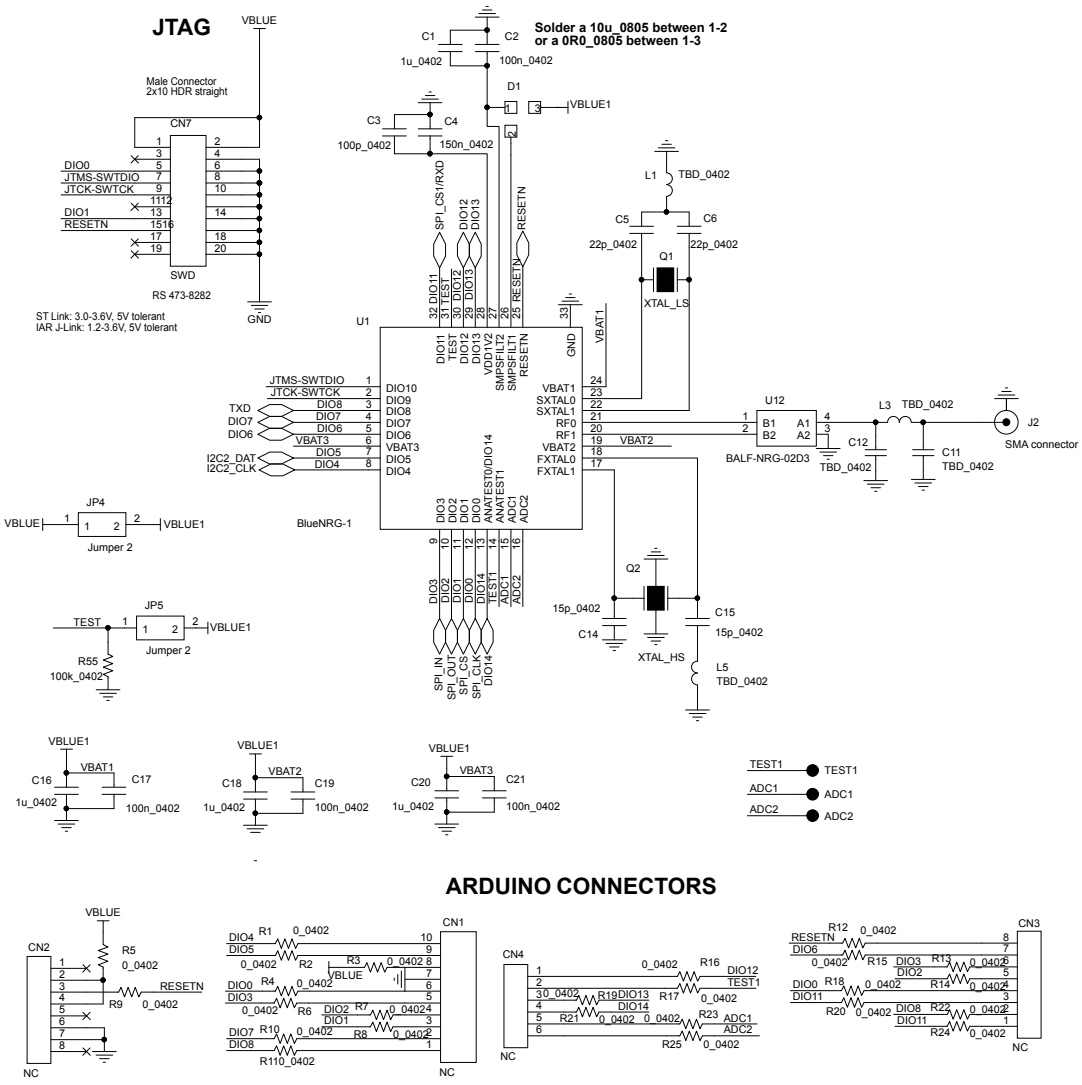


Figure 48. **STEVAL-IDB007V2 - scheme 2**

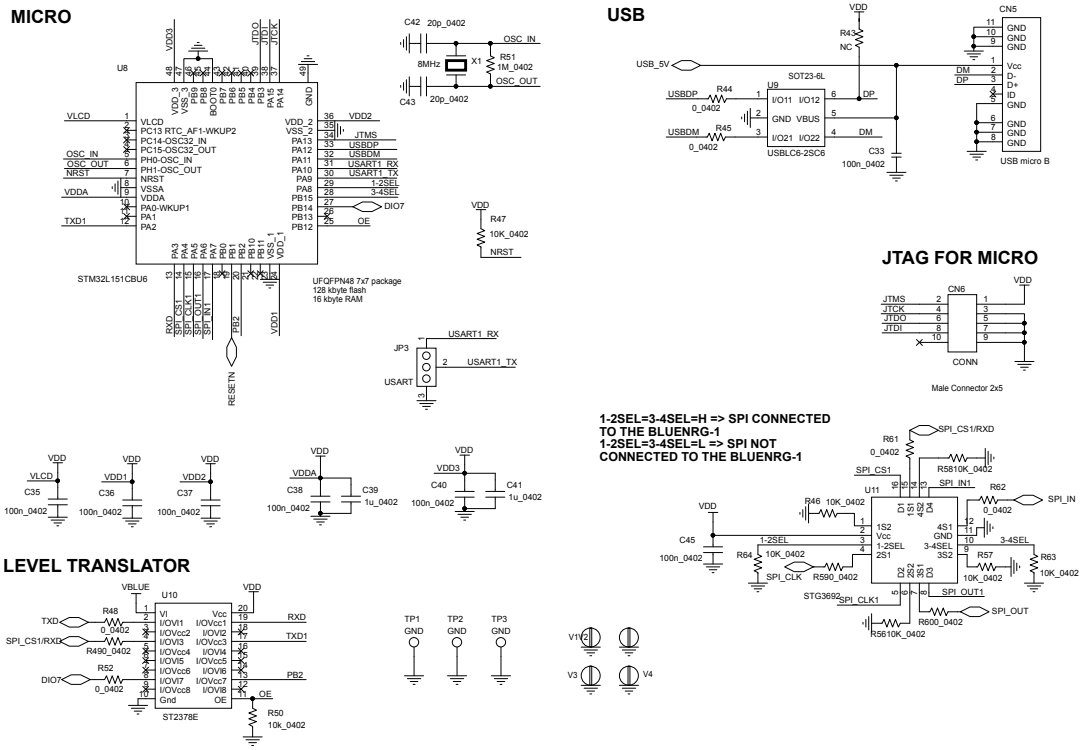
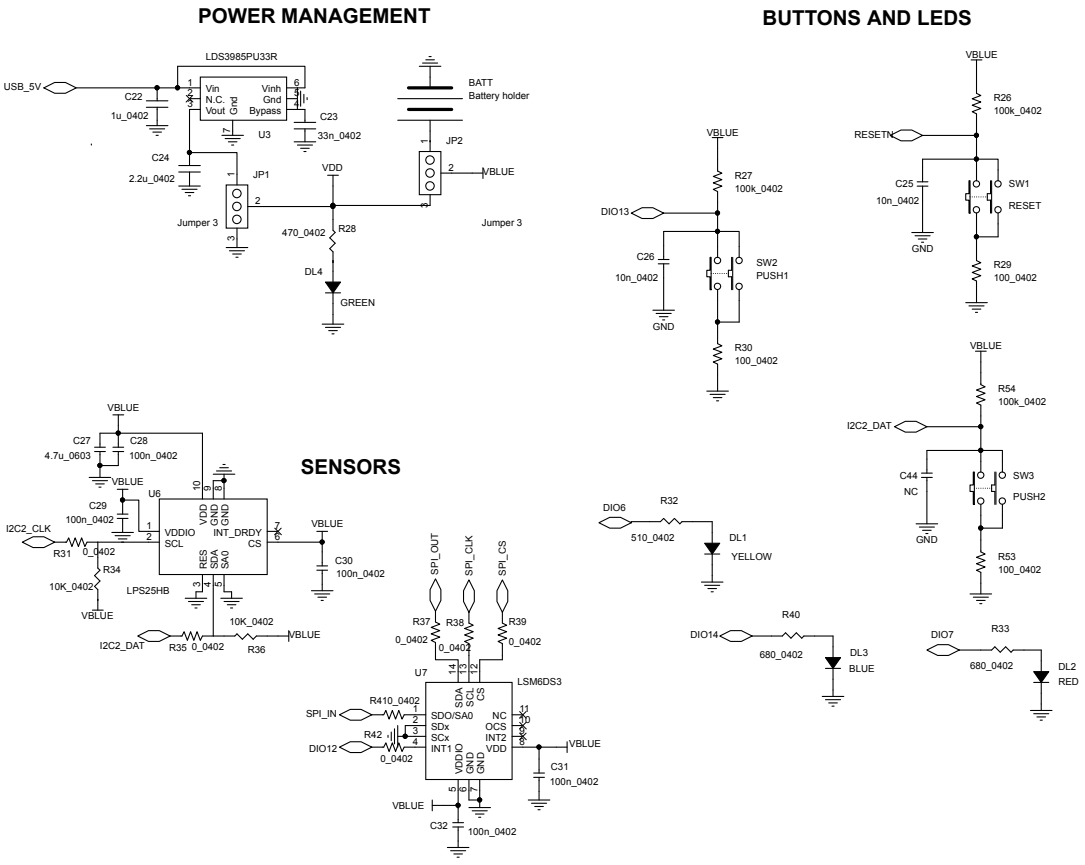


Figure 49. **STEVAL-IDB007V2 - scheme 3**



21 Schematic diagrams for STEVAL-IDB008V2

Figure 50. **STEVAL-IDB008V2 - JTAG**

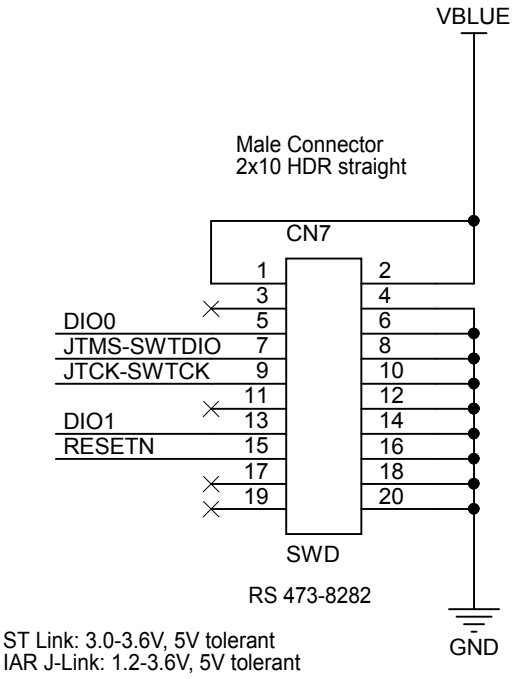


Figure 51. **STEVAL-IDB008V2 - Arduino connection**

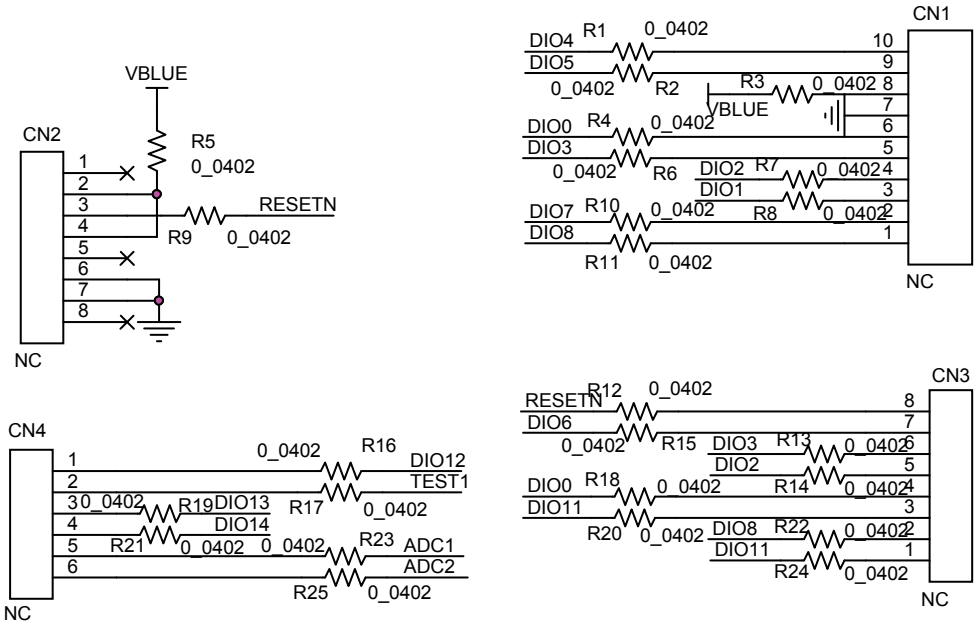


Figure 52. **STEVAL-IDB008V2 circuit schematic**

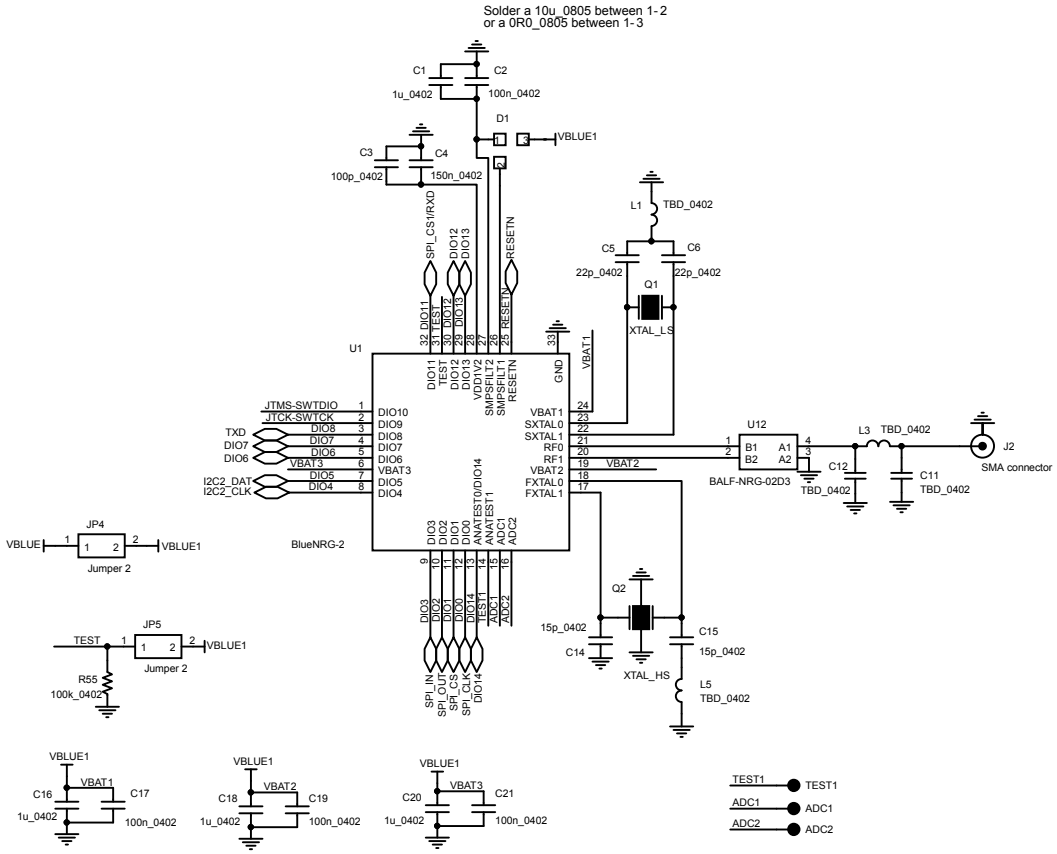


Figure 53. **STEVAL-IDB008V2 - power managements**

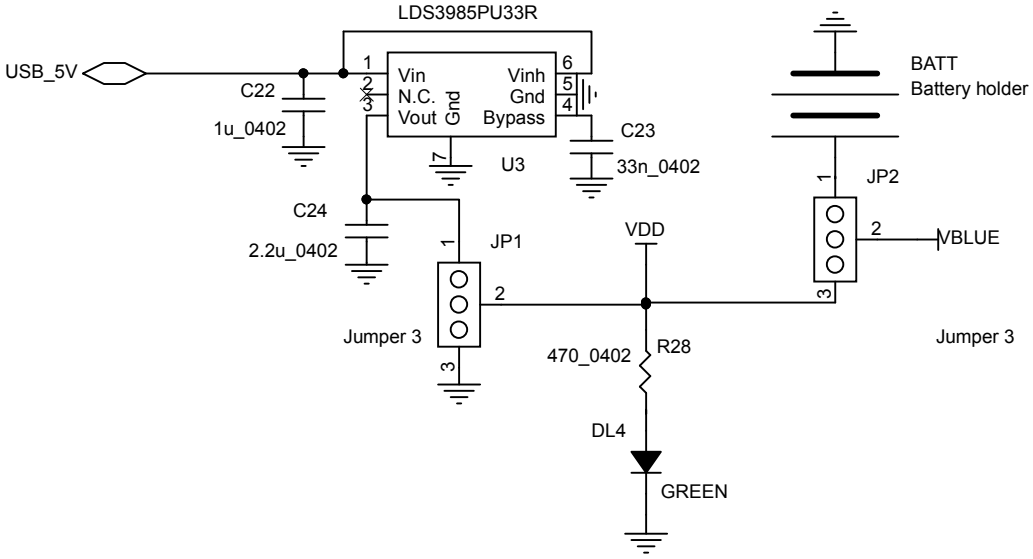


Figure 54. **STEVAL-IDB008V2 - SENSORs**

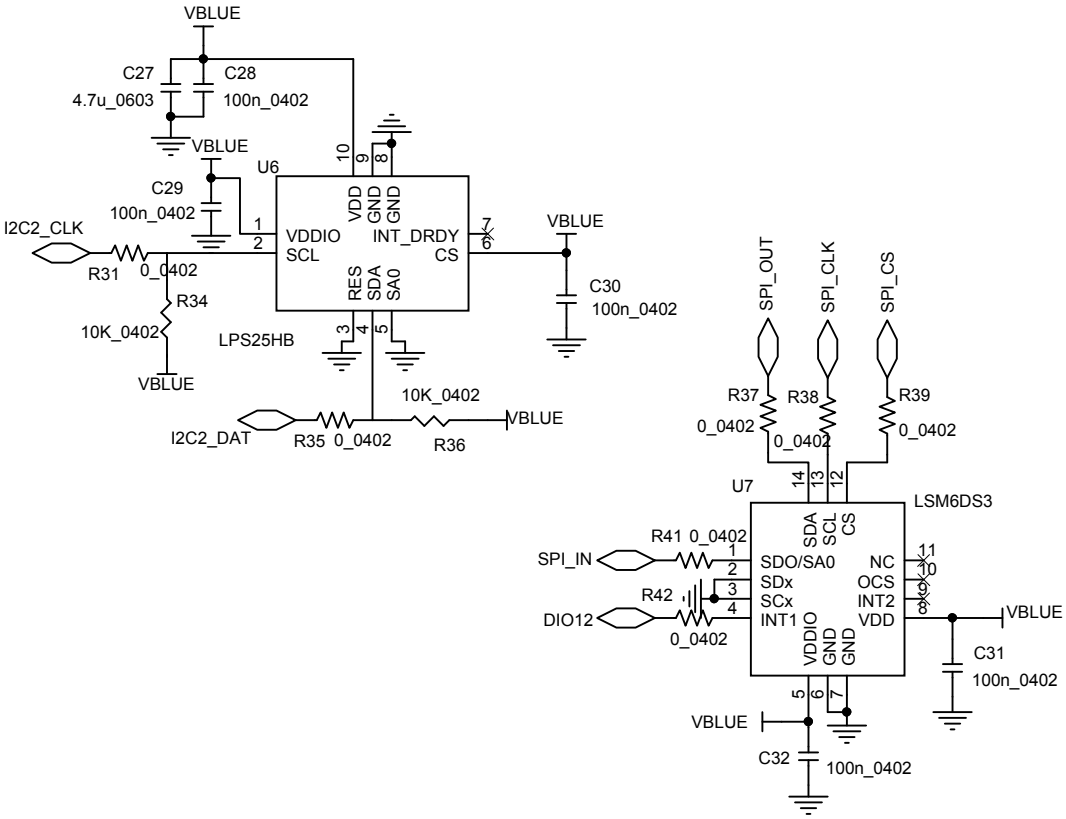


Figure 55. **STEVAL-IDB008V2 - buttons and leds**

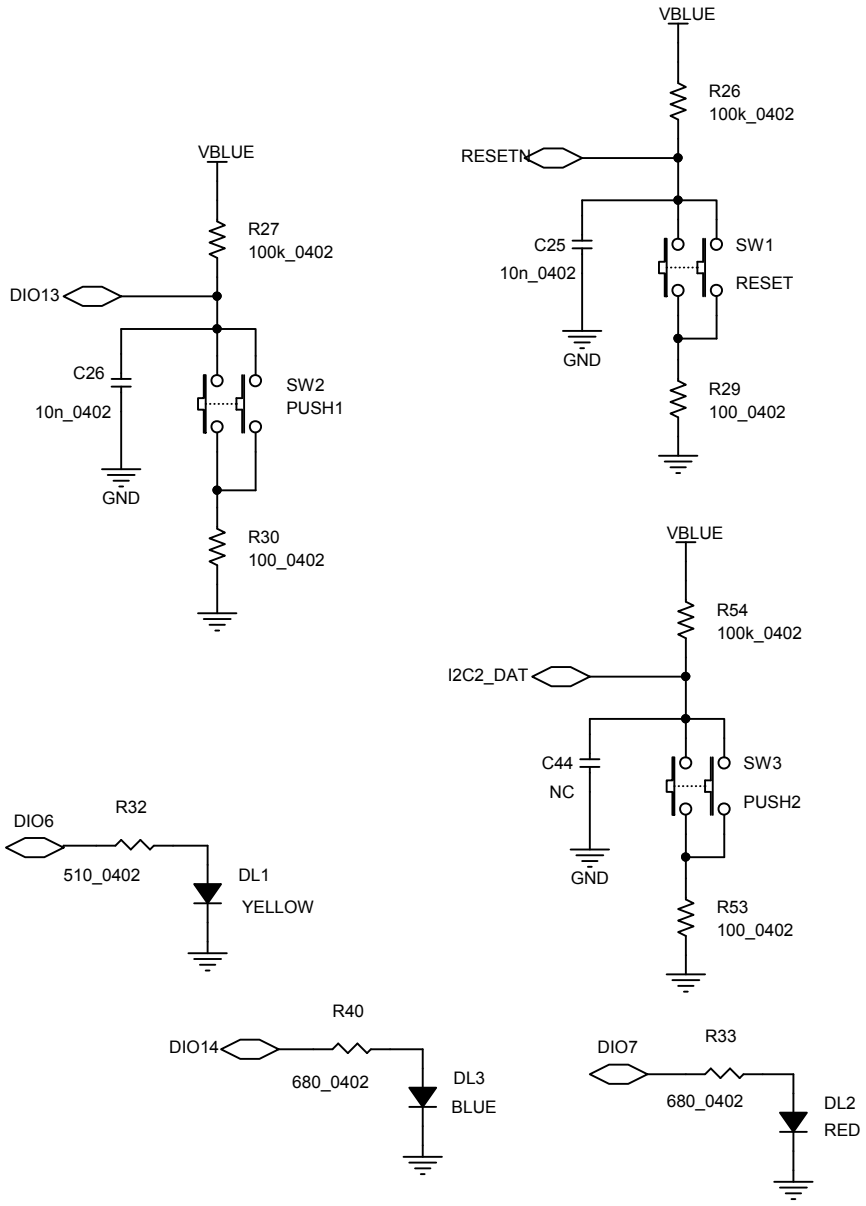
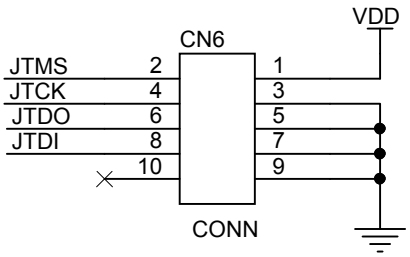


Figure 56. **STEVAL-IDB008V2 - micro**



Male Connector 2x5

Figure 57. **STEVAL-IDB008V2 - USB**

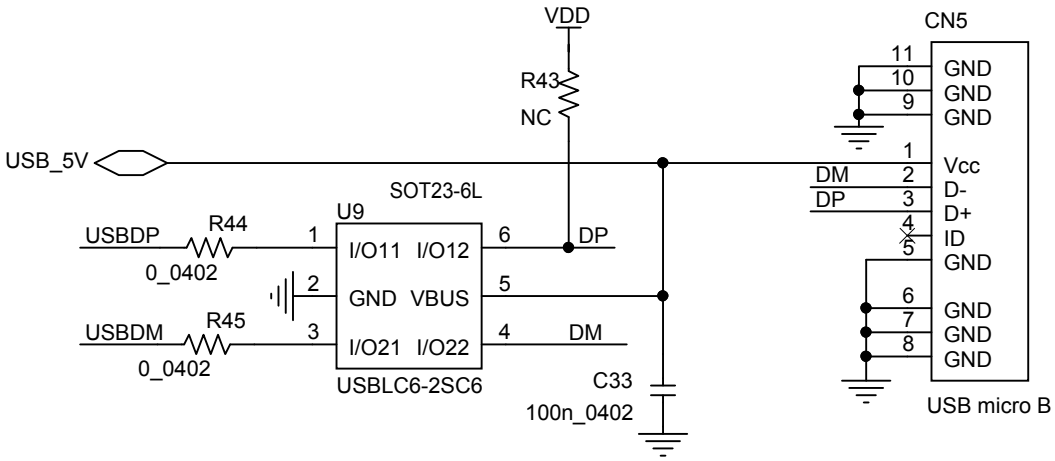
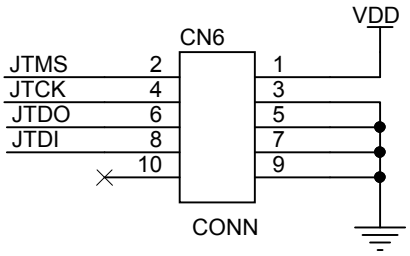


Figure 58. **STEVAL-IDB008V2 - JTAG for micro**



Male Connector 2x5

Figure 59. **STEVAL-IDB008V2 - level translator**

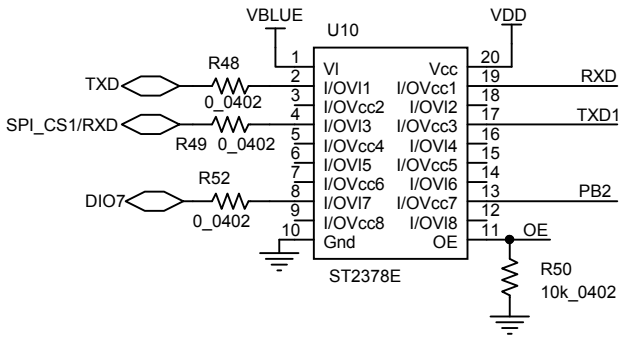
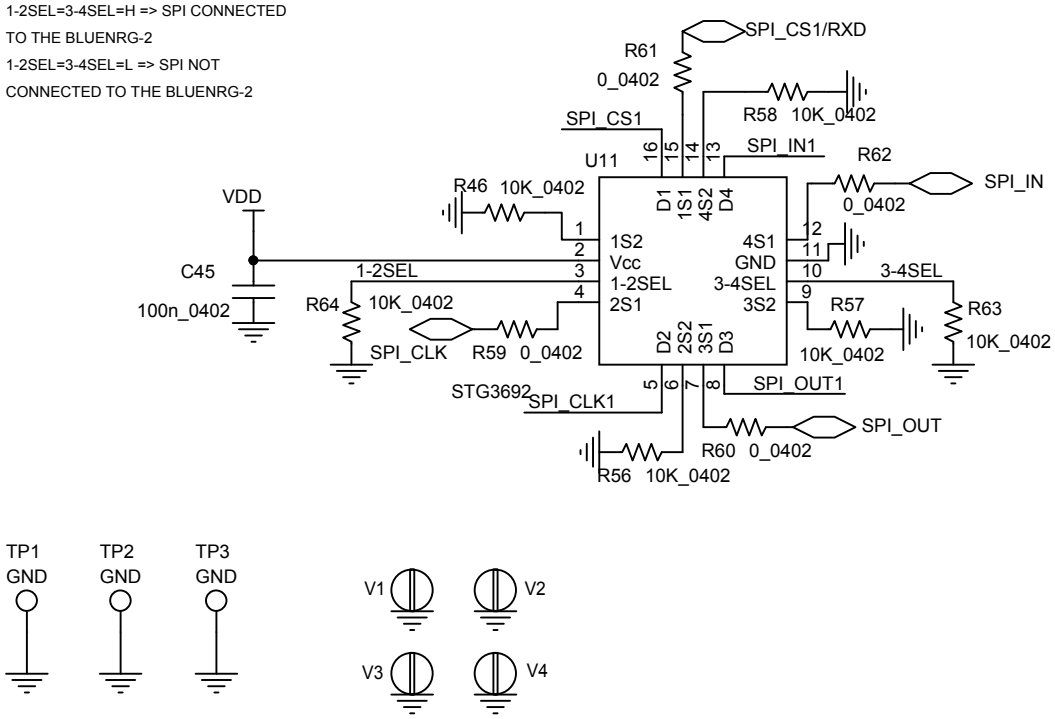


Figure 60. **STEVAL-IDB008V2 - EEPROM**



Revision history

Table 10. **Document revision history**

Date	Version	Changes
06-Jun-2016	1	Initial release.
08-Nov-2016	2	Added *Section 11: "BlueNRG-1 sensor profile central demo"* and description for ADC DMA, PDM and MFT timers.
23-Dec-2016	3	Updated *STEVAL-IDB007V1 development platform* and *STEVAL-IDB007V1 board components*.
27-Jun-2017	4	Updated: *Figure 10: "BLE demonstration and test applications"*, and *Section 18.9: " SPI examples "*. Added: *Section 16: "BLE security demonstration applications"*, *Section 17: "BLE power consumption demo application"*, *Section 18.6: "Public Key Accelerator (PKA) demonstration application"* and *Section 18.7: "RNG examples"*. Added reference to BlueNRG-2 device and related SW components. Add reference to STEVAL-IDB008V1 kit and related schematics pictures.
17-Oct-2017	5	Added reference to BlueNRG-1-V1 DK SW package supporting BLE stack v1.x family.
17-Jan-2018	6	Added references to STEVAL-IDB007V2, STEVAL-IDB008V2 platforms and related schematics.