UM2735

User manual

BlueNRG-LP/BlueNRG-LPS development kits

Introduction

The BlueNRG-LP and BlueNRG-LPS devices are low-power Bluetooth® systems-on-chip, compliant with the Bluetooth® specification and support master, slave, and simultaneous master-and-slave roles.

Both devices also support:

- the Bluetooth® Low Energy data length extension feature;
- 2 Mbps, long range, extended advertising features;
- L2CAP-COS, LE power control;
- periodic advertising and periodic advertising sync features.

Moreover, the BlueNRG-LPS supports the direction finding features: angle of arrival (AoA) and angle of departure (AoD).

The following BlueNRG-LP kits are available:

- STEVAL-IDB011V1 QFN48 package development platform
- STEVAL-IDB011V2 QFN48 package development platform
- STEVAL-IDB010V1 WLCSP49 package development platform

The following BlueNRG-LPS kits are available:

- STEVAL-IDB012V1 QFN32 package development platform
- STEVAL-IDB013V1 WLCSP36 package development platform

The STEVAL-IDB011V1, STEVAL-IDB010V1, STEVAL-IDB011V2, STEVAL-IDB012V1 and STEVAL-IDB013V1 development platforms embed a CMSIS-DAP programming/debugging interface and feature hardware resources for a wide range of application scenarios: sensor data (accelerometer, pressure, and temperature sensor), human interface (buttons and LEDs), digital MEMS microphone (not available on the STEVAL-IDB012V1 and STEVAL-IDB013V1), and serial communication 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-IDB011V1 development platform based on BlueNRG-LP

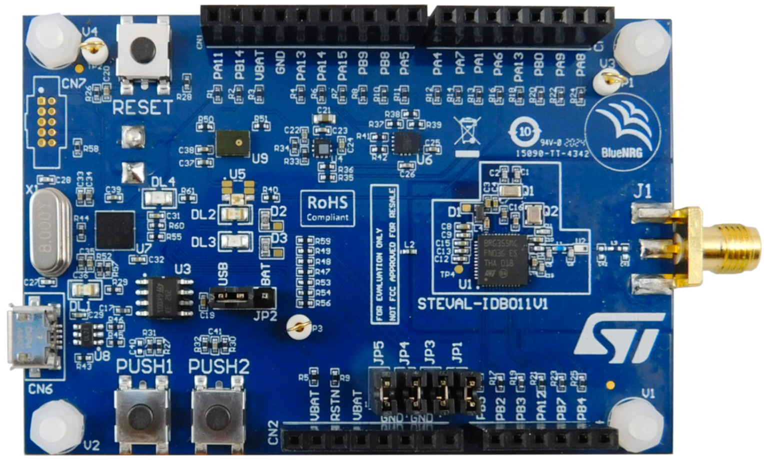


Figure 2. STEVAL-IDB011V2 development platform based on BlueNRG-LP

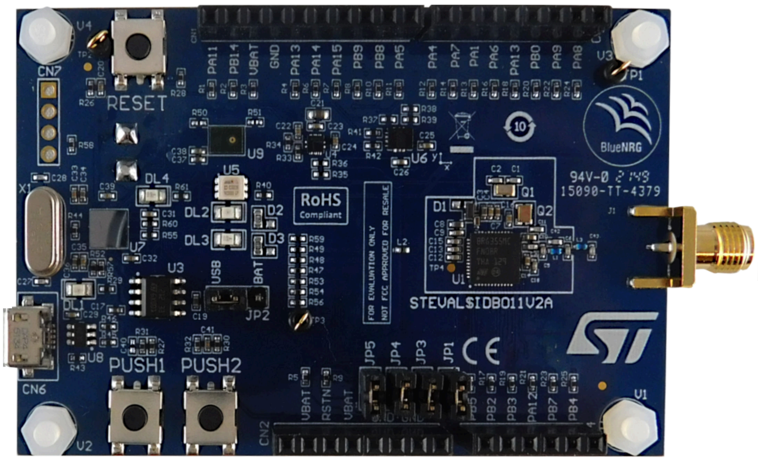


Figure 3. STEVAL-IDB012V1 development platform based on BlueNRG-LPS

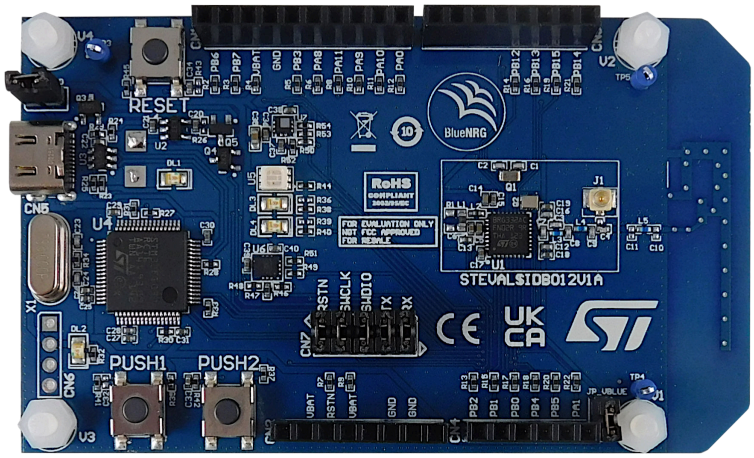


Figure 4. STEVAL-IDB010V1 development platform based on BlueNRG-LP (top view)

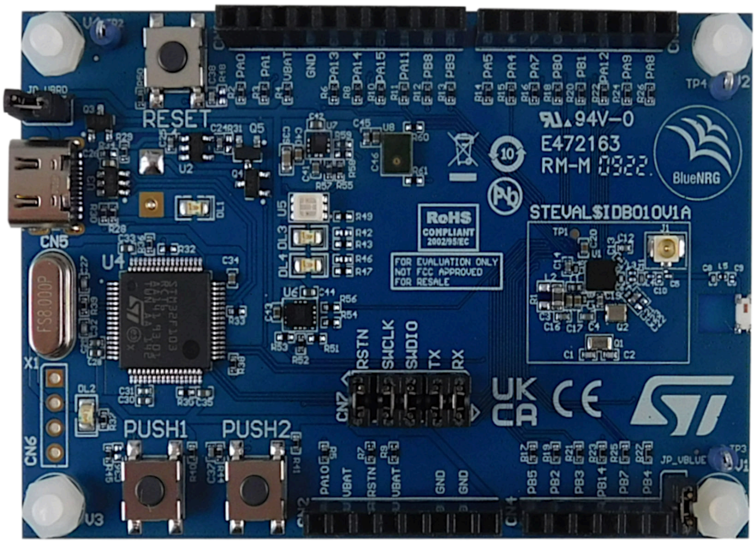


Figure 5. STEVAL-IDB010V1 development platform based on BlueNRG-LP (bottom view)

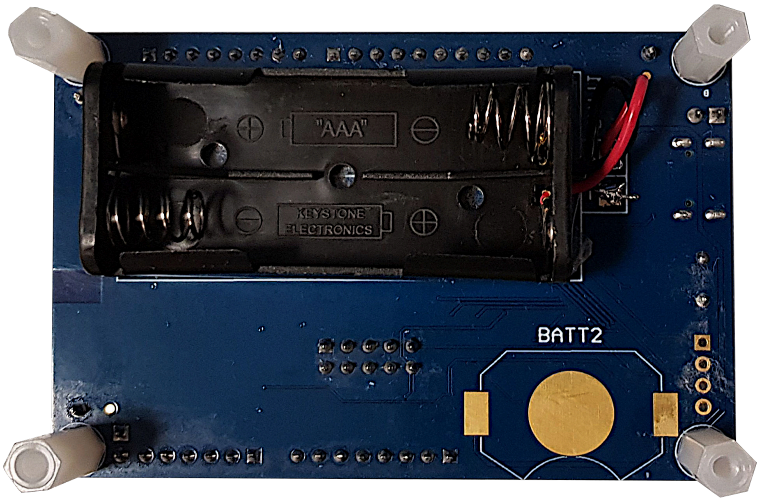
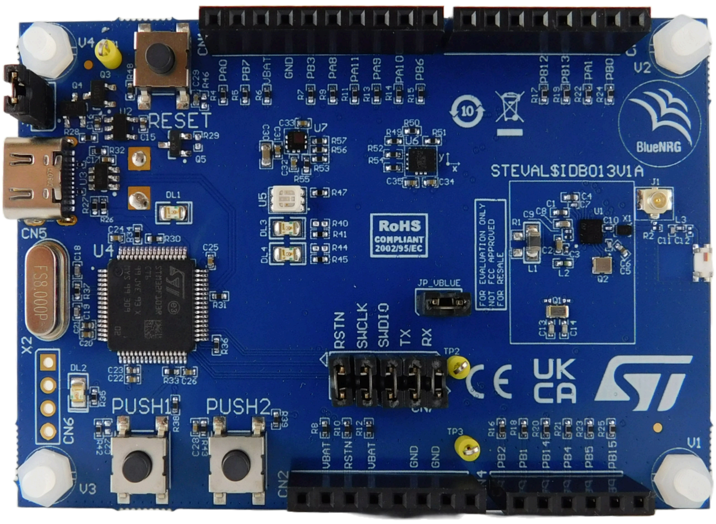


Figure 6. STEVAL-IDB013V1 development platform based on BlueNRG-LPS



1 Getting started

1.1 Safety recommendations

1.1.1 Target audience

These products target users with basic electronics or embedded software development know-how, such as engineers and technicians. These boards are not toys and are not to be used by children.

1.1.2 How to handle the boards

Caution: These products contain bare-printed circuit boards.

Danger:

- *The connection pins on the board might be sharp. Thus, be careful when handling the boards to prevent personal injury.*
- *These boards contain static sensitive devices. To avoid damaging them, handle the boards in an ESD-proof environment.*
- *While powering the boards, do not touch their electric connections with your fingers or anything conductive.*
- *The boards operate at a voltage level that is not dangerous, but their components might be damaged when shorted.*
- *Do not put any liquid on the boards and avoid operating the boards near the water or at a high humidity level.*
- *Do not operate the boards if they are dirty or dusty.*

1.2 Kit contents

The STEVAL-IDB011V1 and STEVAL-IDB011V2 kits include:

- a BlueNRG-LP QFN48 package development platform
- 1/4λ Dipole 1.5 dBi gain 2.4 GHz/Bluetooth SMA antenna

Important: *You can replace the antenna only with one identical (that is an antenna with exactly the same characteristics and features).*

- a micro USB-to-USB-Type A cable

The STEVAL-IDB010V1 kit contains:

- a BlueNRG-LP WLCSP49 package development platform with an on-board chip antenna (2402 MHz to 2480 MHz, 9.2 dBm EIRP)

The STEVAL-IDB012V1 kit contains:

- a BlueNRG-LPS QFN32 package development platform with PCB antenna

The STEVAL-IDB013V1 kit contains:

- a BlueNRG-LPS WLCSP36 package development platform with an on-board chip antenna (2402 MHz to 2480 MHz, 9.08 dBm EIRP)

1.3 System requirements

The BlueNRG-LP/BlueNRG-LPS Navigator and Radio Init Wizard PC applications require:

- PC with Intel® or AMD® processor running:
  – Windows 10 - no driver installation needed
- At least 2 GB of RAM
- USB ports
- Adobe Acrobat Reader 6.0 or later

1.4 BlueNRG-LP/BlueNRG-LPS development kit setup

The STSW-BNRGLP-DK DK software package is available for BlueNRG-LP/BlueNRG-LPS Bluetooth LE stack v3.x family.

After downloading the selected software package, extract en.stsw-bnrglp-dk.zip contents to a temporary directory, launch BlueNRG-LP_LPS DK-x.x.x.x-Setup.exe and follow the on-screen instructions.

Note: EWARM Compiler 8.40.1 or later is required for building the BlueNRG-LP, BlueNRG-LPS DK demonstration applications. The Utility/EWARM_BlueNRG-LP_Flasher_2.1.1 folders must be applied to the local IAR EWARM installation path to add the BlueNRG-LP/BlueNRG-LPS to the list of supported devices.

Note: Keil MDK-ARM toolchain is also supported. The Utility/Keil.STBlueNRG-LP_DFP.3.0.0 pack must be installed on the local Keil MDK-ARM tool to add the BlueNRG-LP/BlueNRG-LPS device to the list of supported devices.

Note: STMicroelectronics WiSE-Studio IDE, GCC toolchain is also supported and related demonstration applications projects are available in the Projects folder of the BlueNRG-LP/BlueNRG-LPS SDK.

2 Hardware description

2.1 STEVAL-IDB011V1 and STEVAL-IDB011V2 board overview

The STEVAL-IDB011V1 and STEVAL-IDB011V2 development kit let you experiment with BlueNRG-LP system-on-chip functions.

These platforms are almost identical. Thus, the same requirements or features apply to both of them. The only differences in the STEVAL-IDB011V2 are: U2 (integrated filter) replacement with discrete components and CN7 replacement with a 1x4 footprint. STEVAL-IDB011V1 and STEVAL-IDB011V2 feature:

- Bluetooth® low energy board based on the BlueNRG-LP Bluetooth low energy system-on-chip (QFN48 package)
- Associated development kit SW package (STSW-BNRGLP-DK) including firmware and documentation
- Bluetooth® low energy compliant, supports master, slave, and simultaneous master-and-slave roles
- Three user LEDs
- Two user buttons
- 3D digital accelerometer and 3D digital gyroscope
- MEMS pressure sensor with embedded temperature sensor
- MEMS audio sensor omnidirectional digital microphone
- Battery holder
- CMSIS-DAP debugger/programmer via micro USB connector
- USB to serial bridge to create an I/O channel with the BlueNRG-LP device
- Jumper to measure BlueNRG-LP current
- RoHS compliant

Figure 7. STEVAL-IDB011V1 board components

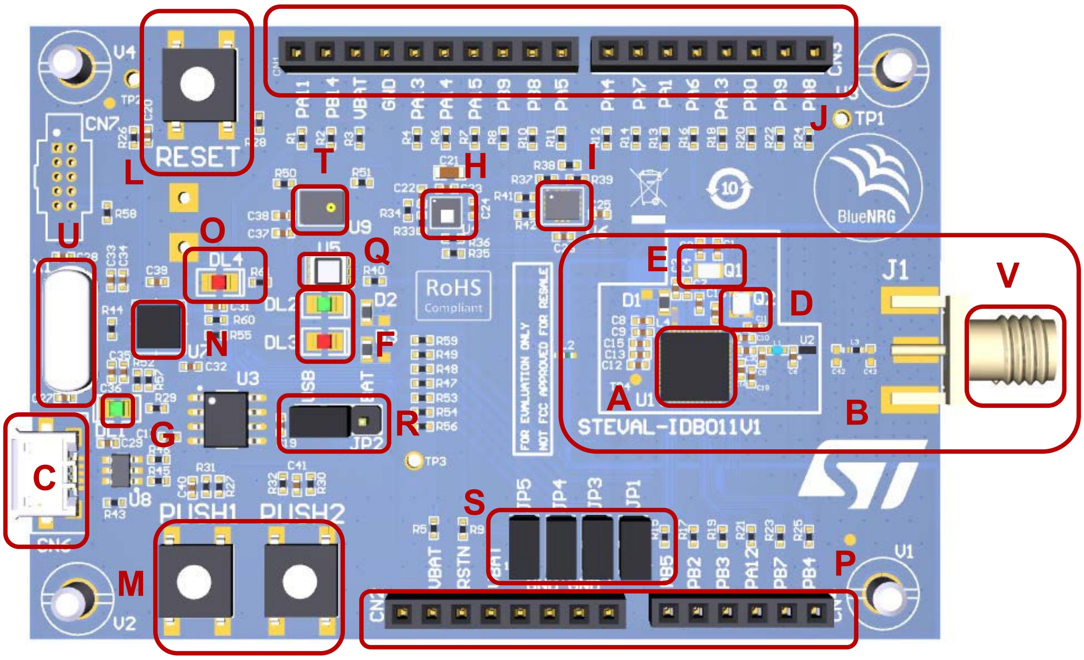


Figure 8. STEVAL-IDB011V2 board components

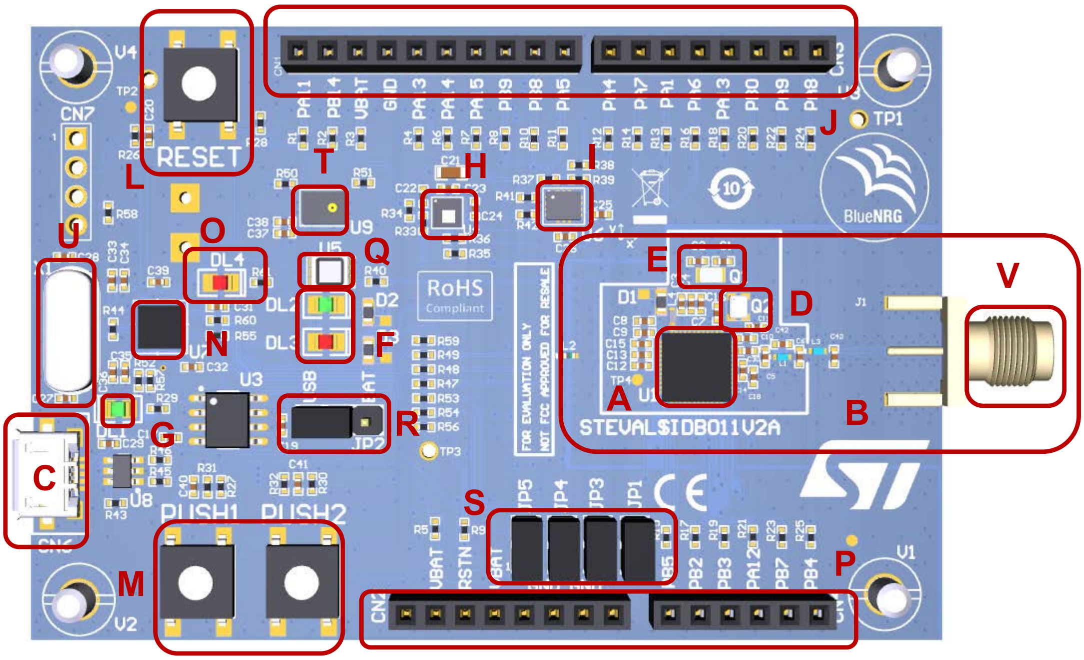


Table 1. STEVAL-IDB011V1 and STEVAL-IDB011V2 boards components description

Region	Description
A	BlueNRG-355MC (BlueNRG-LP, QFN48 package, 256 KB Flash, 64 KB RAM)
C	Micro USB connector for power supply and I/O, and CMSIS-DAP debugger/programmer
L	Reset button
M	Two user buttons
H	LPS22HH MEMS pressure sensor with embedded temperature
I	LSM6DSOX 3D digital accelerometer and 3D digital gyroscope
G	Power LED (DL1)
F	User LEDs (DL2, DL3)
Q	3-color LED (U5)
O	Programmer/debugger and communication activity (LED DL4)
Back of the PCB	Battery holder for two AAA batteries and footprint to solder a battery holder for a CR2032 coin cell
J, P	Two rows of Arduino connectors
N	USB CMSIS-DAP program/debug channel and serial bridge for I/O channel to PC communication (STM32F103xx 64-pin microcontroller)[(1)]
R	Power options (USB, battery)
D	32 MHz high-speed crystal
E	32 kHz low speed crystal
S	Board jumpers
T	MP34DT05-A digital microphone
U	8 MHz crystal
V	SMA connector

1. *Users must not program the STM32.*

2.2 STEVAL-IDB010V1 board overview

The STEVAL-IDB010V1 lets you experiment with the BlueNRG-LP system-on-chip functions.

The STEVAL-IDB010V1 development kit for the BlueNRG-LP in a WLCSP49 package mounts a discrete matching network and a chip antenna.

The on-board U.FL connector grants the RF performance testing. The U.FL connector is not enabled by default and it is not intended to connect external antennas.

The USB Type-C™ connector replaces the micro USB connector for power supply and debugging.

Note: Power and data through a USB Type-C to USB Type-C cable are not supported. Use a USB Type-C to USB Type-A cable instead.

The Tx/Rx interface is exposed on a dedicated connector.

Finally, the power supply from the batteries does not require moving the jumper on the board.

The STEVAL-IDB010V1 features:

- Bluetooth® Low Energy board based on the BlueNRG-LP Bluetooth® Low Energy system-on-chip (WLCSP49 package)
- Associated development kit software package (STSW-BNRGLP-DK) including the firmware and documentation
- Bluetooth® Low Energy compliant, supports master, slave, and simultaneous master-and-slave roles
- Three user LEDs
- Two user buttons
- 3-axis digital accelerometer and 3-axis digital gyroscope
- MEMS pressure sensor with embedded temperature sensor
- MEMS audio sensor omnidirectional digital microphone
- Battery holder
- CMSIS-DAP debugger/programmer via USB Type-C™ connector
- USB to serial bridge to create an I/O channel with the BlueNRG-LP device
- Jumper to measure BlueNRG-LP current
- RoHS compliant

Figure 9. STEVAL-IDB010V1 board components

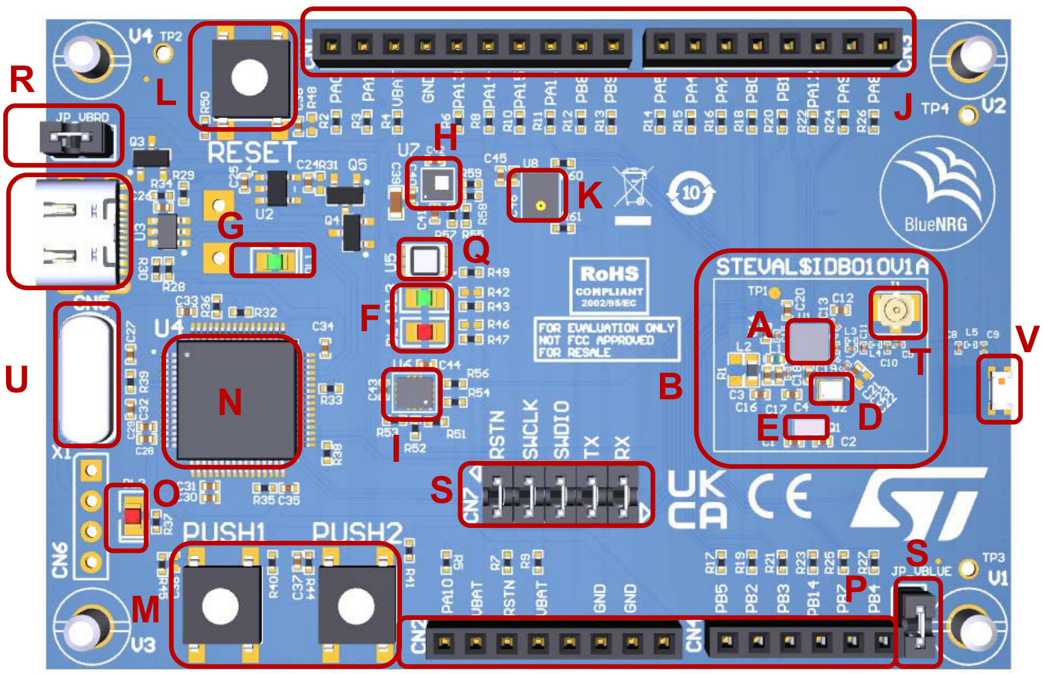


Table 2. **STEVAL-IDB010V1 board components description**

Region	Description
A	BLUENRG-355VC (BlueNRG-LP, WLCSP49 package, 256 KB flash memory, 64 KB RAM)
C	USB Type C™ connector for power supply and I/O, and CMSIS-DAP debugger/programmer
L	Reset button
M	Two user buttons
H	LPS22HH MEMS pressure sensor with embedded temperature
I	LSM6DSOX 3-axis digital accelerometer and 3-axis digital gyroscope
K	MP34DT05-A digital microphone
G	Power LED (DL1)
F	User LEDs (DL3, DL4)
Q	Three-color LED (U5)
O	Programmer/debugger and communication activity (LED DL2)
Back of the PCB	Back of the PCB battery holder for two AAA batteries and footprint to solder a battery holder for a CR2032 coin cell
J, P	Two rows of Arduino connectors
N	USB CMSIS-DAP program/debug channel and serial bridge for I/O channel to PC communication. (STM32F103xx 64-pin microcontroller)[(1)]
R	Power options (USB, battery, external)
D	32 MHz high-speed crystal
E	32 kHz low speed crystal
S	Bluetooth® Low Energy serial interface and jumper for current measurement
U	8 MHz crystal
V	Chip antenna
T	U.FL connector

1. *Do not program the STM32.*

2.3 STEVAL-IDB012V1 and STEVAL-IDB013V1 board overview

The STEVAL-IDB012V1 and STEVAL-IDB013V1 let you experiment with BlueNRG-LPS system-on-chip functions.

The STEVAL-IDB012V1 development kit for the BlueNRG-LPS in QFN32 package mounts a discrete matching network and a PCB antenna.

The STEVAL-IDB013V1 development kit for the BlueNRG-LPS in WLCSP36 package mounts the MLPF-NRG-01D3 IPD and a chip antenna.

The on-board U.FL connector grants RF performance testing. The U.FL connector is not enabled by default and it is not intended to connect external antennas.

The USB Type-C™ connector replaces the micro USB connector for supply and debugging.

Note: Power and data through a USB Type-C to USB Type-C cable are not supported. Use a USB Type-C to USB Type-A cable instead.

The Tx/Rx interface is exposed on a dedicated connector.

Finally, the power supply from the batteries does not require moving the jumper on the board.

The STEVAL-IDB012V1 and STEVAL-IDB013V1 features:

- Bluetooth® Low Energy board based on the BlueNRG-LPS Bluetooth® Low Energy system-on-chip (QFN32 or WLCSP36 package)
- Associated development kit SW package (STSW-BNRGLP-DK) including firmware and documentation
- Bluetooth® Low Energy compliant, supports master, slave, and simultaneous master-and-slave roles
- Three user LEDs

- Two user buttons
- A 3-axis digital accelerometer and a 3-axis digital gyroscope
- A MEMS pressure sensor with embedded temperature sensor
- A battery holder
- A CMSIS-DAP debugger/programmer via USB Type-C™ connector
- A USB-to-serial bridge to create an I/O channel with the BlueNRG-LPS device
- A jumper to measure the BlueNRG-LPS current
- RoHS compliant

Figure 10. **STEVAL-IDB012V1 board components**

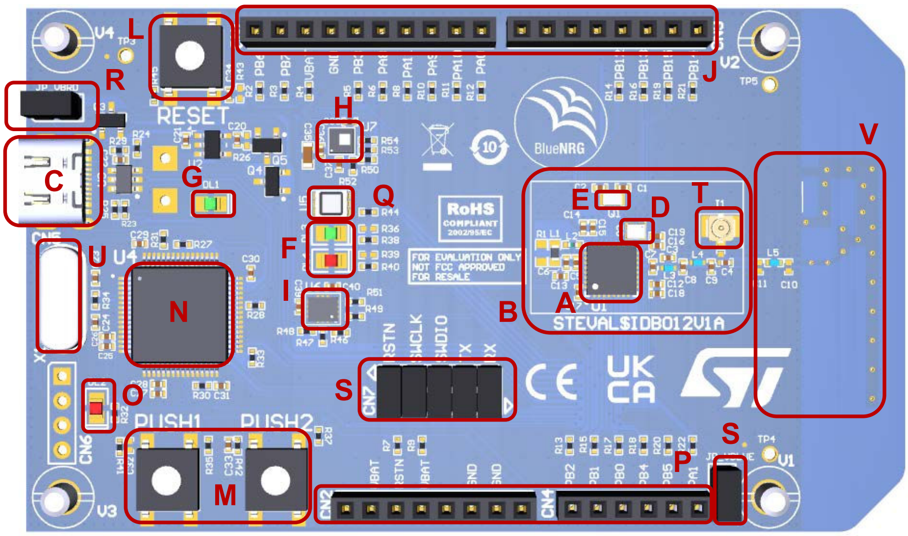


Figure 11. **STEVAL-IDB013V1 board components**

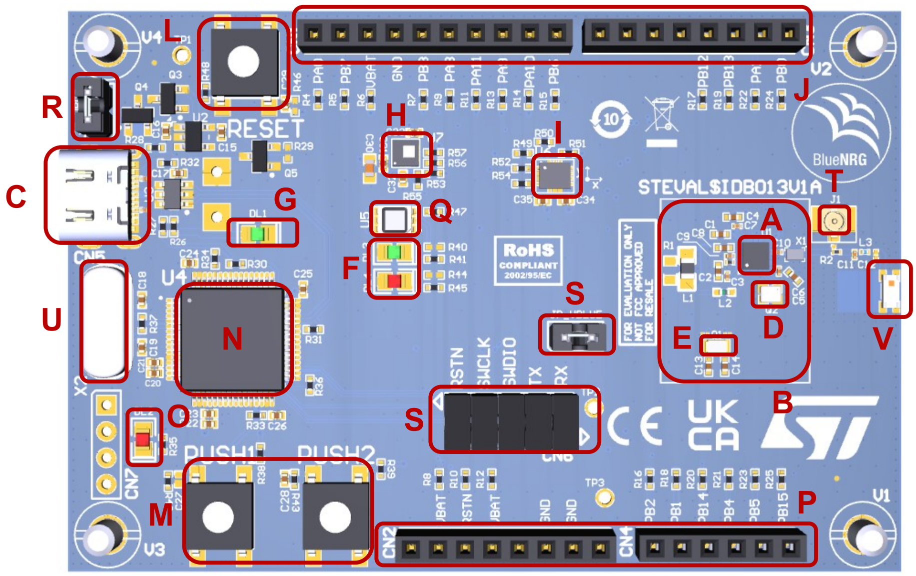


Table 3. **STEVAL-IDB012V1/STEVAL-IDB013V1 board components description**

Region	Description
A	BlueNRG-332AC (BlueNRG-LPS, QFN32 package, 192 KB flash memory, 24 KB RAM) or BlueNRG-332VC (BlueNRG-LPS, WLCSP36 package, 192 KB flash memory, 24 KB RAM)

Region	Description
C	USB Type-C™ connector for power supply and I/O, and CMSIS-DAP debugger/programmer
L	Reset button
M	Two user buttons
H	LPS22HH MEMS pressure sensor with embedded temperature
I	LSM6DSOX 3-axis digital accelerometer and 3-axis digital gyroscope
G	Power LED (DL1)
F	User LEDs (DL3, DL4)
Q	Three-color LED (U5)
O	Programmer/debugger and communication activity (LED DL2)
Back of the PCB	Back of the PCB battery holder for two AAA batteries and footprint to solder a battery holder for a CR2032 coin cell
J,P	Two rows of Arduino connectors
N	A USB CMSIS-DAP program/debug channel and serial bridge for I/O channel to PC communication (STM32F103xx 64-pin microcontroller)(1)
R	Power options (USB, battery, external)
D	32 MHz high-speed crystal
E	32 kHz low speed crystal
S	Bluetooth® Low Energy serial interface and jumper for current measurement
U	8 MHz crystal
V	PCB or chip antenna
T	U.FL connector

1. *Do not program the STM32.*

2.4 BlueNRG-LP SoC connections

The BlueNRG-LP is a very low-power Bluetooth® Low Energy single-mode system-on-chip (Figure 7, Figure 8, and Figure 9–region A) with 256 KB of flash memory, 64 KB of RAM, a 32-bit core Arm® Cortex®-M0+ processor, and several peripherals (ADC, 32 GPIOs, I²C, SPI, I²S, timers, UART, WDG, and RTC).

The microcontroller is connected to various components such as buttons, LEDs, and sensors.

The table below lists the connections and functions available on the STEVAL-IDB011V1 and STEVAL-IDB011V2 development kits.

Table 4. BlueNRG-LP pins description with functions on STEVAL-IDB011V1/STEVAL-IDB011V2

Pin name	Pin no.	Board function										
	QFN48	LEDs	Micro	Buttons	Microphone	LPS22HH	LSM6DSOX	SWD	CN1	CN2	CN3	CN4
PA0	13					I2C_CLK						
PA1	14					I2C_DAT					6	
PA2	15		SWDIO					SWDIO				
PA3	16		SWDIO					SWCLK				
PA4	17										8	
PA5	18								1			
PA6	19	U5									5	
PA7	20										7	

Pin name	Pin no.	Board function										
	QFN48	LEDs	Micro	Buttons	Microphone	LPS22HH	LSM6DSOX	SWD	CN1	CN2	CN3	CN4
PA8	12		UART_RXD								1	
PA9	11		UART_TXD								2	
PA10	10		BOOT	PUSH1								
PA11	9		SPI_CS						10			
PA12	8											4
PA13	7		SPI_CLK				SPI_CLK		6		4	
PA14	6		SPI_MISO				SPI_MISO		5			
PA15	5		SPI_MOSI				SPI_MOSI		4			
PB0	4						INT1				3	
PB1	3				CLK							
PB2	2				DOUT							2
PB3	1											3
PB4	48											6
PB5	47											1
PB6	46			PUSH2								
PB7	45											5
PB8	44	DL2							2			
PB9	43	DL3							3			
PB10	42											
PB11	41											
PB12	30											
PB13	29											
PB14	28								9			
PB15	27						SPI_CS					
RSTN	38		RSTN	RESET						3		
GND	49								7	7, 6		

Table 5. **BlueNRG-LP pins description with functions on STEVAL-IDB010V1**

Pin Name	Pin no.	Board function										
	WLCSP49	LEDs	Micro	Buttons	Microphone	LPS22HH	LSM6DSOX	SWD	CN1	CN2	CN3	CN4
PA0	F6					I2C_CLK			10			
PA1	G6					I2C_DAT			9			
PA2	E5		SWDIO					SWDIO				
PA3	E4		SWCLK					SWCLK				
PA4	E3										7	

Pin Name	Pin no.	Board function										
	WLCSP49	LEDs	Micro	Buttons	Microphone	LPS22HH	LSM6DSOX	SWD	CN1	CN2	CN3	CN4
PA5	F5										8	
PA7	G5										6	
PA8	E7		UART_RXD								1	
PA9	D7		UART_TXD								2	
PA10	E6		BOOT	PUSH1						1		
PA11	D3		SPI_CS						3			
PA12	D6										3	
PA13	D5		SPI_CLK				SPI_CLK		6			
PA14	C7		SPI_MISO				SPI_MISO		5			
PA15	C6		SPI_MOSI				SPI_MOSI		4			
PB0	C5						INT1				5	
PB1	C4				CLK						4	
PB2	B7				DOUT							
PB3	B6											
PB4	B5											6
PB5	B4											1
PB6	B3			PUSH2								
PB7	B2											5
PB8	D4	DL3							2			
PB9	C3	DL4							1			
PB12	C2											
PB13	D2											
PB14	E2											4
PB15	F2						SPI_CS					
RSTN	A5		RSTN	RESET						3		
GND	F3, F7, A7, B1, F4, G3, A3, G1								7	2,6		

Region B in Figure 7, Figure 8, and Figure 9 includes the following main components:

- BlueNRG-LP low-power system-on-chip
- 32 MHz high frequency crystal
- 32 kHz low frequency crystal for the lowest power consumption
- SMA or U.FL connector

2.5 BlueNRG-LPS SoC connections

The BlueNRG-LPS is a very low-power Bluetooth® Low Energy single-mode system-on-chip (Figure 10. STEVAL-IDB012V1 board components and Figure 11. STEVAL-IDB013V1 board components region A) with 192 KB of flash memory, 24 KB of RAM, a 32-bit core Arm® Cortex®-M0+ processor, and several peripherals (ADC, 20 GPIOs, I²C, SPI, I²S, timers, UART, WDG, and RTC).

The microcontroller is connected to various components such as buttons, LEDs, and sensors.

The table below lists the connections and functions available on the STEVAL-IDB012V1 and STEVAL-IDB013V1 development kits.

Table 6. BlueNRG-LPS pins description with functions on STEVAL-IDB012V1

Pin name	Pin no.	Board function									
	QFN32	LEDs	Micro	Buttons	LPS22HH	LSM6DSOX	SWD	CN1	CN2	CN3	CN4
PA0	8				I2C_CLK			1			
PA1	7		UART_TX								6
PA2	6		SWDIO				SWDIO				
PA3	5		SWCLK				SWCLK				
PA8	9					SPI_MISO		5			
PA9	10							3			
PA10	11		BOOT	PUSH1				2			
PA11	12					SPI_MOSI		4			
PB0	4		UART_RX								3
PB1	3	U5									2
PB2	2	DL4									1
PB3	1					SPI_CLK		6			
PB4	31	DL3									4
PB5	30			PUSH2							5
PB6	23							10			
PB7	22				I2C_DAT			9			
PB12	21									4	
PB13	20									3	
PB14	19					INT1				2	
PB15	18					SPI_CS				1	
RSTN	29		RSTN	RESET					3		
GND	33							7	2, 6		

Table 7. BlueNRG-LPS pins description with functions on STEVAL-IDB013V1

Pin name	Pin no.	Board function									
	WLCSP36	LEDs	Micro	Buttons	LPS22HH	LSM6DSOX	SWD	CN1	CN2	CN3	CN4
PA0	H7				I2C_CLK			10			
PA1	F7		UART_TX							2	
PA2	G6		SWDIO				SWDIO				
PA3	H5		SWCLK				SWCLK				
PA8	M7					SPI_MISO		5			
PA9	J6							3			
PA10	K5		BOOT	PUSH1				2			
PA11	J4					SPI_MOSI		4			
PB0	D7		UART_RX							1	
PB1	E6	U5									2
PB2	D5	DL4									1
PB3	C6					SPI_CLK		6			
PB4	F5	DL3									4

Pin name	Pin no.	Board function									
	WLCSP36	LEDs	Micro	Buttons	LPS22HH	LSM6DSOX	SWD	CN1	CN2	CN3	CN4
PB5	G4			PUSH2							5
PB6	C4					SPI_CS		1			
PB7	E4				I2C_DAT			9			
PB12	C2									4	
PB13	D3									3	
PB14	E2					INT1					3
PB15	F3										6
RSTN	B5		RSTN	RESET					3		
GND	A2, B7, H1, K7, L6, L4							7	2, 6		

Region B in Figure 10 and Figure 11. STEVAL-IBD013V1 board components includes the following main components:

- BlueNRG-LPS low-power system-on-chip
- 32 MHz high frequency crystal
- 32 kHz low frequency crystal for the lowest power consumption
- U.FL connector

2.6 Power supply

DL1 green LED (Figure 7, Figure 8, Figure 9, Figure 10 and Figure 11, region G) signals the board is being powered 5 Vdc, 50 mA from the micro USB connector CN6 (Figure 7 and Figure 8, region C) or the USB Type-C™ connector CN5 (Figure 9, Figure 10 and Figure 11, region C).

The board can be also powered via:

- 3 Vdc, 50 mA from 2 x 1.5 Vdc AAA not rechargeable alkaline batteries on the BATT1 region at the rear of the board

Danger: *Do not replace the battery with an incorrect type, as there is a serious risk of fire or explosion.*

Important: *Check the polarity of the batteries before inserting them in the battery holder.*

- a CR2032 coin cell battery (if the coin battery support is soldered on the BATT2 region at the rear of the board)
- an external DC power supply [1.7 V to 3.6 V]

Table 8. STEVAL-IDB011V1/STEVAL-IDB011V2 kit platform power supply modes

Power supply mode	JP2 settings	Description
1 - USB	USB position	USB supply through CN6 connector (Figure 7, region C)
2 - Battery	BAT position	The supply voltage must be provided through the batteries (at the rear of the board).
3–External DC power supply	Removed	The supply voltage must be provided through JP2 pin 2. The USB connection is not needed

Table 9. STEVAL-IDB010V1/STEVAL-IDB012V1/STEVAL-IDB013V1 kit platform power supply modes

Power supply mode	JP_VBRD	Description
1 - USB	No change required	USB supply through CN5 (Figure 10, region C)
2 - Battery	No change required	The supply voltage must be provided through the batteries (at the rear of the board).

Power supply mode	JP_VBRD	Description
3 - External DC power supply	JP_VBRD removed	The supply voltage must be provided through JP_VBRD pin 1. The USB connection is not needed.

2.7 Physical layer testing

The data traffic on the USB connector (C) could degrade the test results when a physical layer is tested. In such cases, it is recommended to supply the board externally and use a USB to serial converter (for example, FTDI cable) for data transfer.

2.8 Circuit protection

Power supply is always provided to ensure circuit protection. For example, a 2.0 USB port allows drawing a maximum of 500 mA.

The development board draws a maximum current of 50 mA.

2.9 Operating temperature

Use the equipment at room temperature (25°C).

2.10 Main transient voltage

The equipment is protected against a maximum of 1500 V_{PEAK} main transient voltage.

2.11 Jumpers

The following jumpers are available (Figure 7, Figure 8, Figure 9, and Figure 10, region S).

Table 10. STEVAL-IDB011V1/STEVAL-IDB011V2 kit platform jumpers

Jumper	Description
JP1	It provides the voltage to the BlueNRG-LP circuit. It must be adapted and can be used for BlueNRG-LP current measurements.
JP2	It is a switch between two power domains: • BAT position - to provide power from battery holder • USB position - to provide power from USB connector
JP3	It connects the BlueNRG-LP BLE_SWCLK pin with the USB_CMSISDAP SWCLK pin. It must be adapted.
JP4	It connects the BlueNRG-LP BLE_SWDIO pin with the USB_CMSISDAP SWDIO pin. It must be adapted.
JP5	It connects the BlueNRG-LP BLE_RSTN pin a pin of the USB_CMSISDAP. It must be adapted.

Table 11. STEVAL-IDB010V1/STEVAL-IDB012V1/STEVAL-IDB013V1 kit platform jumpers

Jumper	Description
JP_VBLUE	It provides the voltage to the BlueNRG-LP/BlueNRG-LPS circuit. It can be used for BlueNRG-LP/BlueNRG-LPS current measurements.
JP_VBRD	It provides the power supply coming from the on-board regulator or the battery.
CN7.1-2	It connects the BlueNRG-LP/BlueNRG-LPS UART Tx pin to the UART Rx pin of CMSIS DAP-Link.
CN7.3-4	It connects the BlueNRG-LP/BlueNRG-LPS UART Rx pin to the UART Tx pin of CMSIS DAP-Link.
CN7.5-6	It connects the BlueNRG-LP/BlueNRG-LPS SWDIO pin to the SWDIO pin of CMSIS DAP-Link.
CN7.7-8	It connects the BlueNRG-LP/BlueNRG-LPS SWCLK pin to the SWCLK pin of CMSIS DAP-Link.
CN7.9-10	It connects the BlueNRG-LP/BlueNRG-LPS RSTN pin to CMSIS DAP-Link.

2.12 Sensors

The following sensors are available on the platforms:

1. An LPS22HH (Figure 7, Figure 8, Figure 9, Figure 10 and Figure 11 region H) piezo resistive absolute pressure sensor with embedded temperature 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 LSM6DSOX (Figure 7, Figure 8, Figure 9, Figure 10 and Figure 11, region I) 3D digital accelerometer and 3D digital gyroscope which communicates via SPI interface. One line for interrupt is also connected
3. An MP34DT05-A MEMS audio sensor omnidirectional digital microphone connected to the BlueNRG-LP PDM port.

Note: The digital microphone is not present on the STEVAL-IDB012V1/STEVAL-IDB013V1 kit.

2.13 Extension connector

BlueNRG-LP/BlueNRG-LPS signal test points are shared on two Arduino connector rows: CN1, CN3 (Figure 7, Figure 8, Figure 9, Figure 10 and Figure 11 region J), and CN2, CN4 (Figure 7, Figure 8, Figure 9, Figure 10 and Figure 11, region P).

According to JP2 settings on the STEVAL-IDB011V1/STEVAL-IDB011V2 kits and JP_VBRD settings on the STEVAL-IDB010V1/STEVAL-IDB012V1/STEVAL-IDB013V1 kits (see Table 8 and Table 9) the power on the CN2.2 and CN1.8 pins is delivered by:

- the on-board voltage regulator, that is, U3 on the STEVAL-IDB011V1/STEVAL-IDB011V2 and U2 on the STEVAL-IDB010V1/STEVAL-IDB012V1/STEVAL-IDB013V1. Refer to LDL212D33R and LDK120M33R datasheets, respectively
- the batteries (the typical capacity of one alkaline battery is 1.3–1.8 Wh)
- the external DC power supply

2.14 Push buttons

The board has one user button to reset the microcontroller (Figure 7, Figure 8, Figure 9, Figure 10 and Figure 11, region L) and two further buttons for application purposes (region N).

2.15 LEDs

LEDs DL1 (green power LED), DL2 (green), DL3 (red), U5 (blue), and DL4 (USB_CMSISDAP activity red LED) are available on the STEVAL-IDB011V1/STEVAL-IDB011V2 kits (Figure 7 and Figure 8, regions G, F,Q, and O).

LEDs DL1 (green power LED), DL3 (green), DL4 (red), U5 (blue), and DL2 (USB_CMSISDAP activity red LED) are available on the STEVAL-IDB010V1/STEVAL-IDB012V1/STEVAL-IDB013V1 kit (Figure 9, Figure 10 and Figure 11, regions G, F,Q, and O).

2.16 CMSIS-DAP and Virtual COM

The most important features of the STM32F103xx microcontroller (Figure 7, Figure 8, Figure 9, Figure 10 and Figure 11, region N) are:

- CMSIS-DAP debugging/programming capability through the USB micro connector
- USB-to-serial bridge providing an I/O communication channel with the BlueNRG-LP/BlueNRG-LPS device (to interface with a USB host device as a PC)
- drag and drop capability to program the BlueNRG-LP/BlueNRG-LPS

Note: Users must not program the on-board STM32 microcontroller. ST provides a preprogrammed firmware image (USB_CMSISDAP.hex) to interface the BlueNRG-LP/BlueNRG-LPS with a USB host device with the highlighted features.

2.16.1 Virtual COM port driver setup for Windows

In Windows 10, no installation is required.

2.16.2 System functionality checks

To check if the system is ready to use, follow the steps below.

Step 1. Check whether the CMSIS-DAP device is present in the Windows Device Manager.

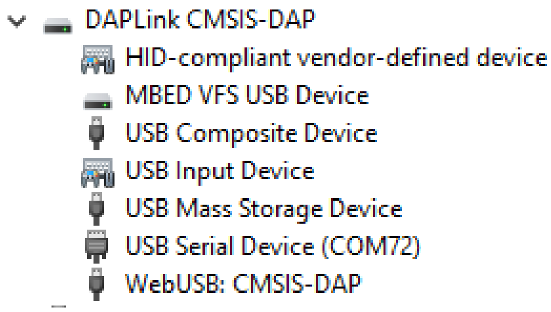


Figure 12. Windows Device Manager - CMSIS-DAP

Note: The composite device (WebUSB: CMSIS-DAP) installation is not required, as this functionality is not used.

Step 2. Check under [**Devices and drives**] whether the ST IDB011VX mass storage device is present (if the board is powered by the USB cable and connected to a PC). The same applies to IDB012VX and IDB010VX kits.

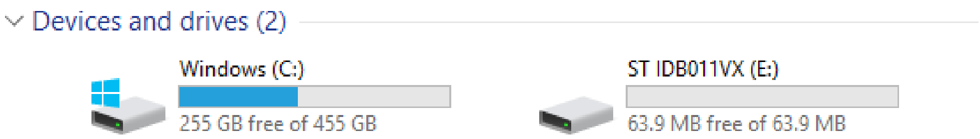


Figure 13. ST IDB011VX mass storage device

Step 3. Check whether the DL1 LED is on (if the board is powered by USB).

2.16.3 USB_CMSISDAP programming/debugging feature

The STEVAL-IDB011V1, STEVAL-IDB011V2, STEVAL-IDB010V1, STEVAL-IDB012V1, and STEVAL-IDB013V1 enable this on-board programming/debugging feature (USB_CMSISDAP). To use it, you have to choose CMSIS DAP as debugger/programmer in IAR EWARM, Keil® µVision, or WiSE-Studio GCC toolchain development environments.

Figure 14. IAR EWARM project - debugger option

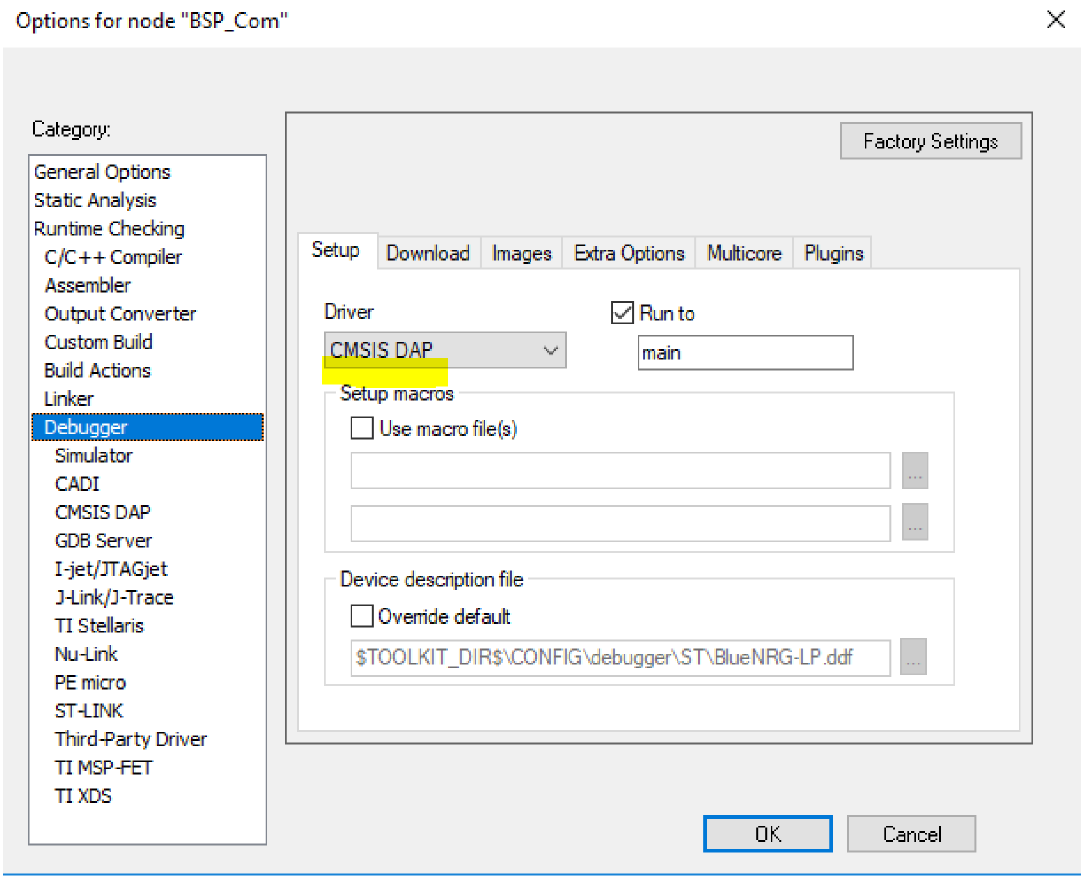


Figure 15. **Keil® µVision project - debugger option**

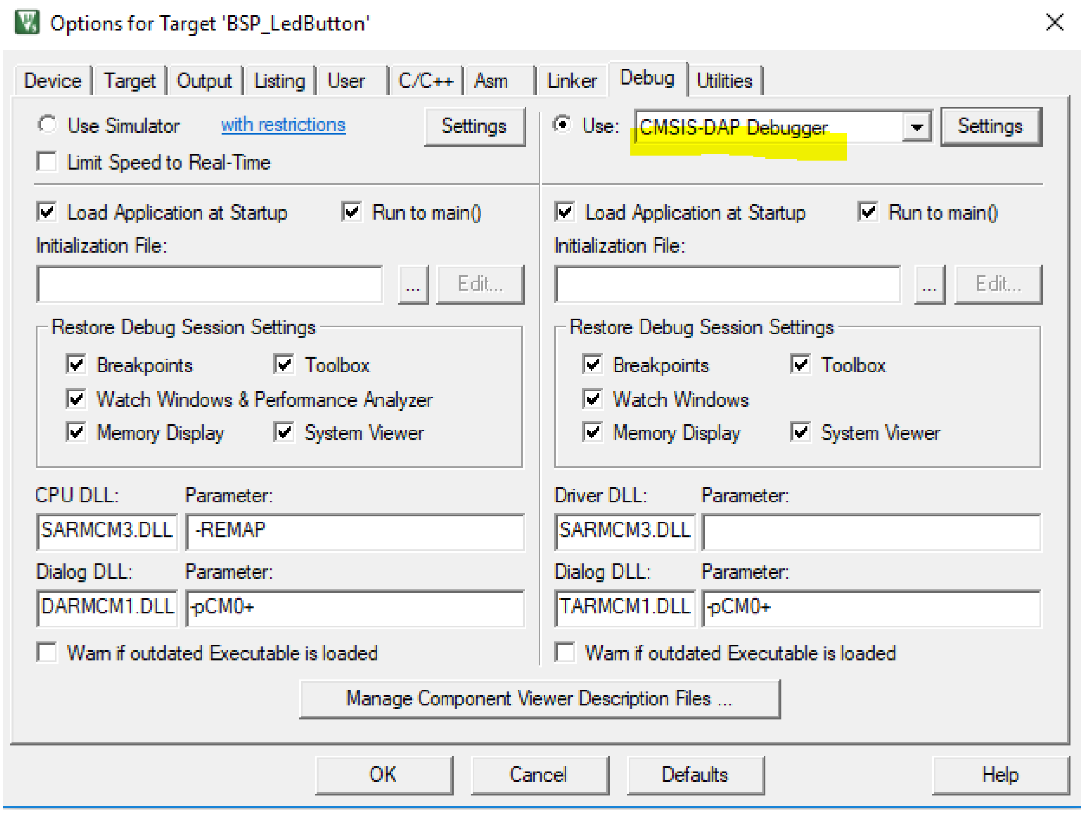
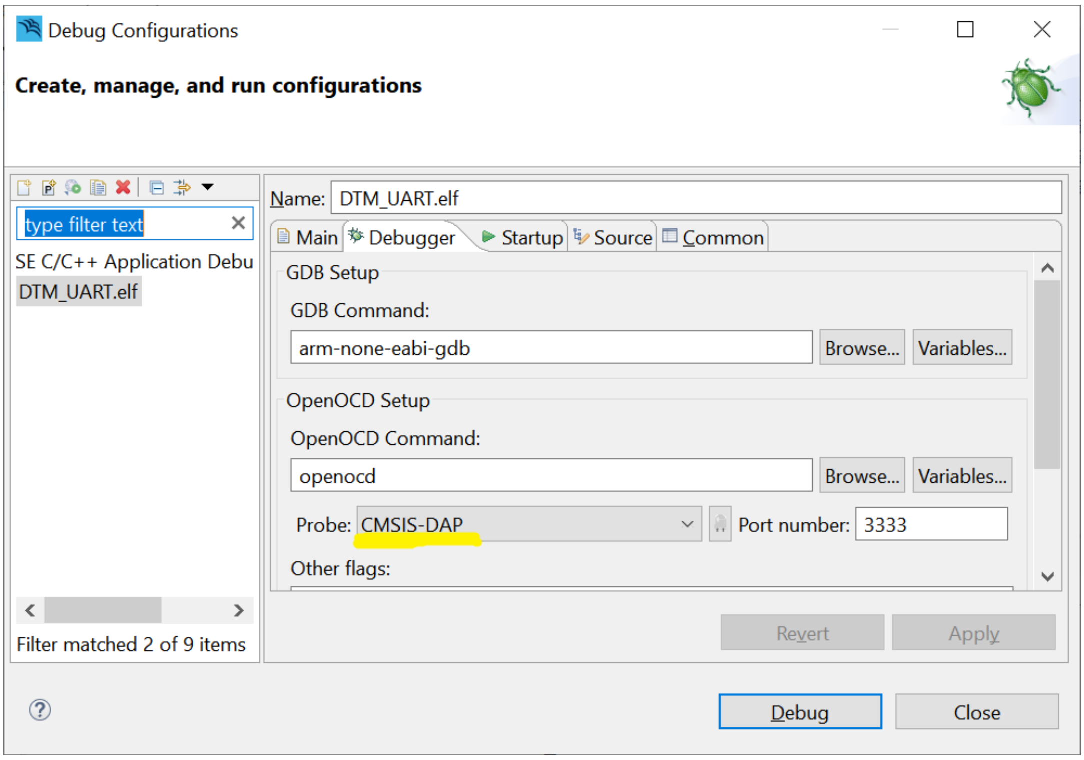


Figure 16. **WiSE-Studio project - debugger option**



You can load a binary image (.bin or .hex) to the BlueNRG-LP/BlueNRG-LPS by copying and pasting (or dragging and dropping) the binary file into the ST IDB01XVX mass storage device.

2.16.4 USB_CMSISDAP firmware update

If an updated version of the USB_CMSISDAP_LP and USB_CMSISDAP_LPS firmware is released, respectively for BlueNRG-LP and BlueNRG-LPS boards, you should follow the procedure below for firmware update.

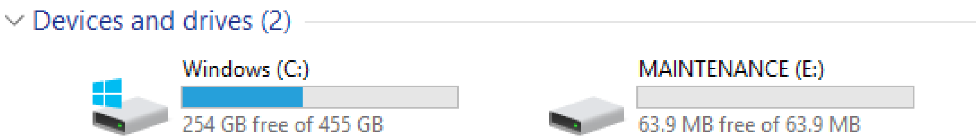
Step 1. Unplug the USB cable (if plugged).

Step 2. Press and hold the [**RESET**] button.

Step 3. Plug the USB cable.

Step 4. Release the [**RESET**] button.
A new mass storage device ([**MAINTENANCE**]) appears.

Figure 17. **USB_CMSISDAP firmware - MAINTENANCE mass storage device**



Step 5. Copy and paste the new binary image into the [**MAINTENANCE**] mass storage device.

Step 6. At the end of the operation, unplug and then plug the USB cable again to start the board up.

2.17 BlueNRG-LP/BlueNRG-LPS programming and debugging

To program and debug the BlueNRG-LP/BlueNRG-LPS embedded in the STEVAL-IDB011V1, STEVAL-IDB011V2, STEVAL-IDB010V1, STEVAL-IDB012V1, and STEVAL-IDB013V1 boards, you can use an external SWD programmer/debugger tool after removing:

- JP3, JP4, and JP5 jumpers from the STEVAL-IDB011V1 and STEVAL-IDB011V2 kit platforms and connecting the SWD tool to the board as listed in the table below

Table 12. **External SWD and STEVAL-IDB011V1/STEVAL-IDB011V2 pin connections**

SWD pins	Kit platform pins
SWDIO	JP4 pin 1
SWCLK	JP3 pin 1
NRST	JP5 pin 1
GND	GND
Target VCC	VBAT

- CN7.5-6, CN7.7-8, and CN7.9-10 jumpers from the STEVAL-IDB010V1, STEVAL-IDB012V1 and STEVAL-IDB013V1 kit platform and connecting the SWD tool to the board as listed in the table below

Table 13. **External SWD and STEVAL-IDB010V1/STEVAL-IDB012V1/STEVAL-IDB013V1 pin connections**

SWD pins	Kit platform pins
SWDIO	CN7.6
SWCLK	CN7.8
NRST	CN7.10
GND	GND
Target VCC	VBAT

Remember: *Before using the supported IDE toolchains, select the related SWD programmer/debugger tool from the IAR EWARM project (Option/Debugger), from the Keil® µVision project (Option/Debug), and from the WiSE-Studio GCC project (Run/Debug configurations).*

The CMSIS-DAP debugging/programming capability via the USB connector can also be used to program/debug a BlueNRG-LP/BlueNRG-LPS device on a different board.

- On STEVAL-IDB011V1 and STEVAL-IDB011V2, remove JP3, JP4, and JP5 jumpers. Then, connect the BlueNRG-LP kit platform to the board SWD pins, GND, and target VCC as follows:

Table 14. STEVAL-IDB011V1/STEVAL-IDB011V2 kit platform and user board pin connection

BlueNRG-LP kit platform pins	User board SWD pins
JP4 pin 2	SWDIO
JP3 pin 2	SWCLK
JP5 pin 2	NRST
GND	GND
VBAT	Target VCC

- On the STEVAL-IDB010V1/STEVAL-IDB012V1/STEVAL-IDB013V1, remove CN7.5-6, CN7.7-8, and CN7.9-10 jumpers. Then, connect the BlueNRG-LP/BlueNRG-LPS kit platform to the board SWD pins, GND, and target VCC as follows:

Table 15. STEVAL-IDB010V1/STEVAL-IDB012V1/STEVAL-IDB013V1 kit platform and user board pin connection

Kit platform pins	User board SWD pins
CN7.5	SWDIO
CN7.7	SWCLK
CN7.9	NRST
GND	GND
VBAT	Target VCC

Finally, connect the BlueNRG-LP/BlueNRG-LPS kit platform USB to a PC USB port to start programming and debugging the BlueNRG-LP/BlueNRG-LPS device on your board.

2.18 Current measurements

To monitor the BlueNRG-LP/BlueNRG-LPS power consumption, you must remove:

- the jumper from JP1 and insert an ammeter between the connector pins 1 and 2 on STEVAL-IDB011V1/STEVAL-IDB011V2 kits
- the jumper JP_VBLUE and insert an ammeter between the connector pins 1 and 2 on the STEVAL-IDB010V1/STEVAL-IDB012V1/STEVAL-IDB013V1 kit

Since the BlueNRG-LP/BlueNRG-LPS power consumption is usually very low, an accurate instrument in the range of few micro amps is recommended.

2.19 Hardware setup for STEVAL-IDB011V1/IDB011V2 kits

Step 1. Connect an antenna to the SMA connector.

Step 2. Configure the STEVAL-IDB011V1/STEVAL-IDB011V2 board to USB power supply mode as per Table 8.

Step 3. Connect the STEVAL-IDB011V1/STEVAL-IDB011V2 board to a PC via USB cable (CN6 connector).

Step 4. Check whether the DL1 power indication LED is on.

2.20 Hardware setup for STEVAL-IDB010V1/STEVAL-IDB012V1/STEVAL-IDB013V1 kit

Step 1. Configure the STEVAL-IDB010V1/STEVAL-IDB012V1/STEVAL-IDB013V1 board to USB power supply mode as per Table 9.

Step 2. Connect the STEVAL-IDB010V1/STEVAL-IDB012V1/STEVAL-IDB013V1 board to a PC via USB cable (CN5 connector).

Step 3. Check whether the DL1 power indication LED is on.

3 BlueNRG-LP/BlueNRG-LPS Navigator

The BlueNRG-LP/BlueNRG-LPS Navigator GUIs let you select and run demonstration applications without additional hardware. It provides access to the following DK software package components:

- BlueNRG-LP/BlueNRG-LPS Bluetooth® Low Energy demonstration applications
- BlueNRG-LP/BlueNRG-LPS peripheral driver examples (LL, HAL & MIX)
- BlueNRG-LP/BlueNRG-LPS 2.4 GHz radio proprietary examples
- BlueNRG-LP/BlueNRG-LPS development kits
- Release notes
- License files

With the BlueNRG-LP/BlueNRG-LPS DK Navigator, you can directly download and run the selected prebuilt application binary image (Bluetooth® Low Energy examples or peripheral driver example) on the BlueNRG-LP/ BlueNRG-LPS platforms without a JTAG interface.

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

You can run the utility through the BlueNRG-LP Navigator icon under: [**Start**]>[**ST BlueNRG-LP_LPS DK X.X.X**]>[**BlueNRG-LP Navigator**], [**BlueNRG-LPS Navigator**].

Note: In this section, the BlueNRG-LP Navigator is shown as an example. Similar concepts are applicable to the BlueNRG-LPS Navigator.

Figure 18. BlueNRG-LP Navigator

3.1 Demonstration applications

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

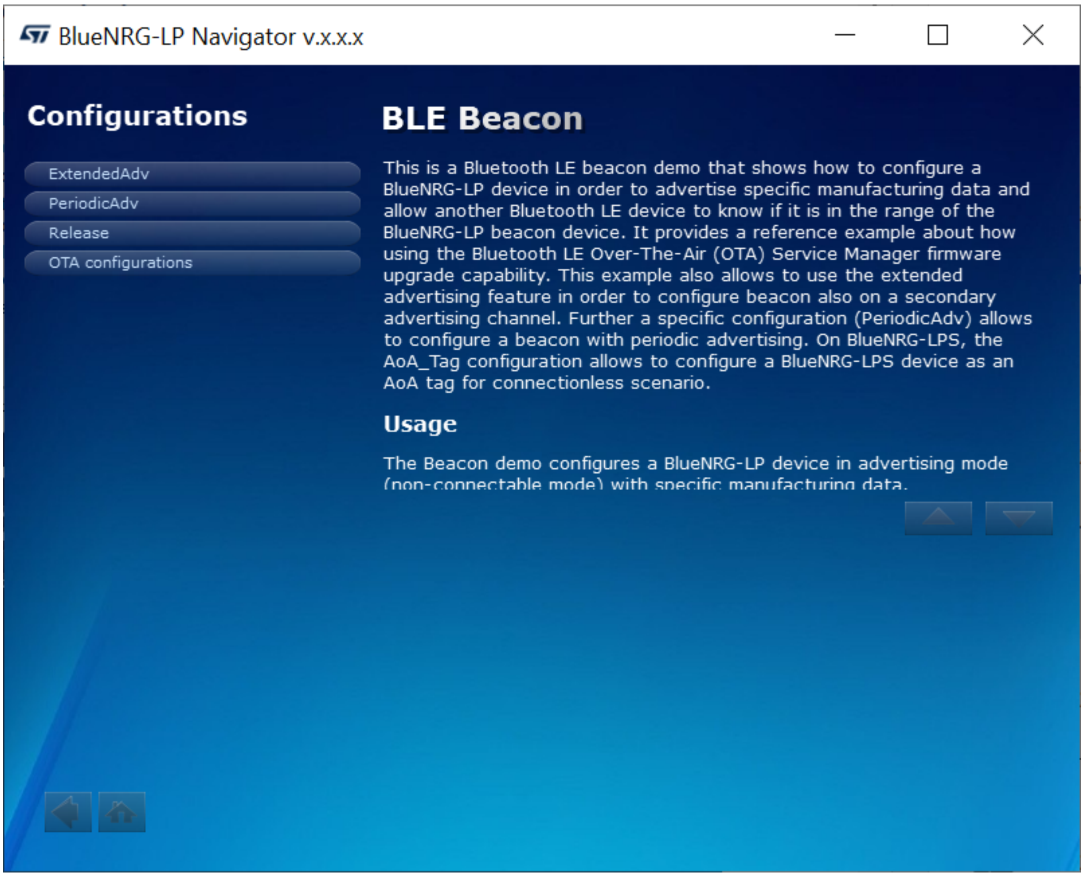
- application settings (if applicable)
- application description
- application hardware related information (e.g., LED signals, jumper configurations, etc.)

The following functions are also available for each application:

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

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

Figure 19. **BlueNRG-LP Navigator - BLE Beacon application**

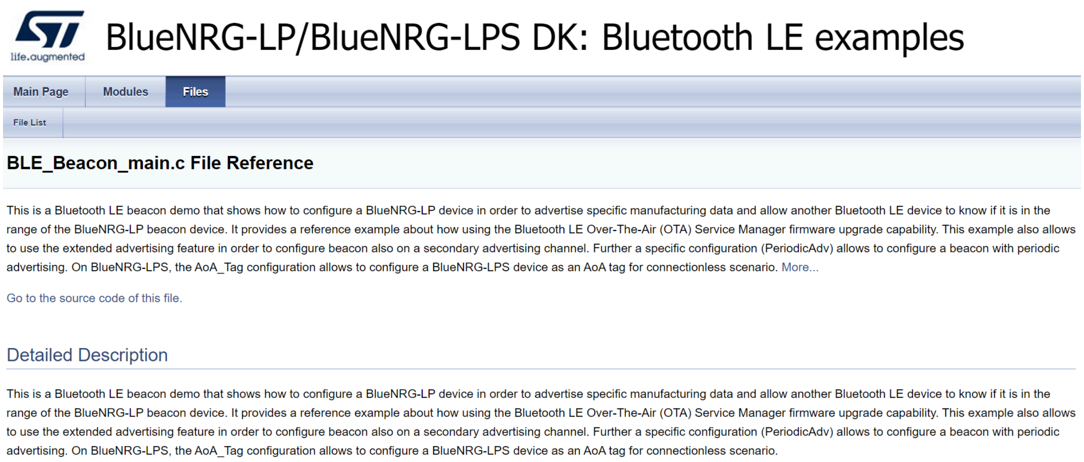


When the BlueNRG-LP platform is connected to your PC USB port, you can select the [**Flash & Run**] tab on the application window to download and run the available pre-built application binary image on the BlueNRG-LP platform.

Figure 20. **BlueNRG-LP Navigator - BLE Beacon Flash programming**

Selecting the [**Doc**] tab opens the related html documentation.

Figure 21. **BLE Beacon documentation**



BlueNRG-LP/BlueNRG-LPS DK: Bluetooth LE examples

Main Page | Modules | Files

File List

BLE_Beacon_main.c File Reference

This is a Bluetooth LE beacon demo that shows how to configure a BlueNRG-LP device in order to advertise specific manufacturing data and allow another Bluetooth LE device to know if it is in the range of the BlueNRG-LP beacon device. It provides a reference example about how using the Bluetooth LE Over-The-Air (OTA) Service Manager firmware upgrade capability. This example also allows to use the extended advertising feature in order to configure beacon also on a secondary advertising channel. Further a specific configuration (PeriodicAdv) allows to configure a beacon with periodic advertising. On BlueNRG-LPS, the AoA_Tag configuration allows to configure a BlueNRG-LPS device as an AoA tag for connectionless scenario. More...

Go to the source code of this file.

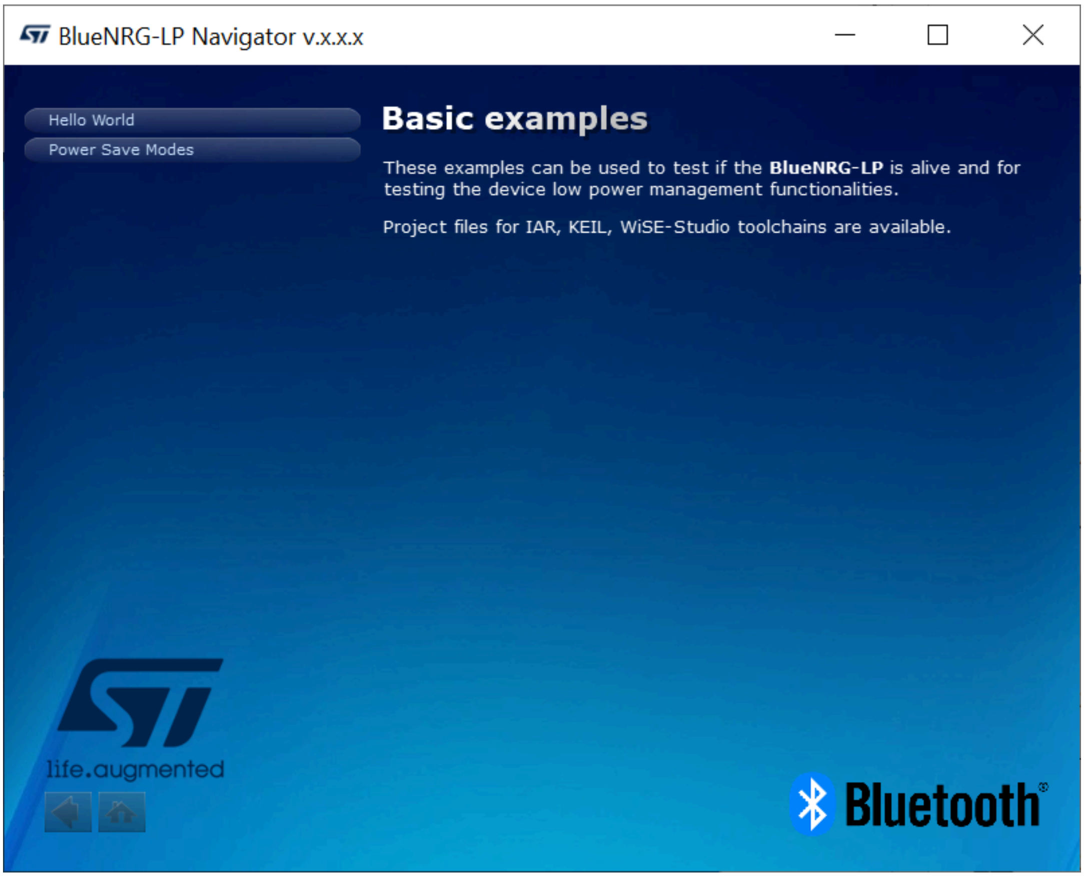
Detailed Description

This is a Bluetooth LE beacon demo that shows how to configure a BlueNRG-LP device in order to advertise specific manufacturing data and allow another Bluetooth LE device to know if it is in the range of the BlueNRG-LP beacon device. It provides a reference example about how using the Bluetooth LE Over-The-Air (OTA) Service Manager firmware upgrade capability. This example also allows to use the extended advertising feature in order to configure beacon also on a secondary advertising channel. Further a specific configuration (PeriodicAdv) allows to configure a beacon with periodic advertising. On BlueNRG-LPS, the AoA_Tag configuration allows to configure a BlueNRG-LPS device as an AoA tag for connectionless scenario.

3.2 Basic examples

This page lists some basic sample applications to verify some of the BlueNRG-LP device modes (alive, sleeping, wake-up).

Figure 22. BlueNRG-LP Navigator - BlueNRG-LP basic examples



3.3 BLE demonstration and test applications

This page lists all the available Bluetooth LE demonstration applications in the DK software package. The applications provide usage examples of the Bluetooth LE stack features for the BlueNRG-LP device.

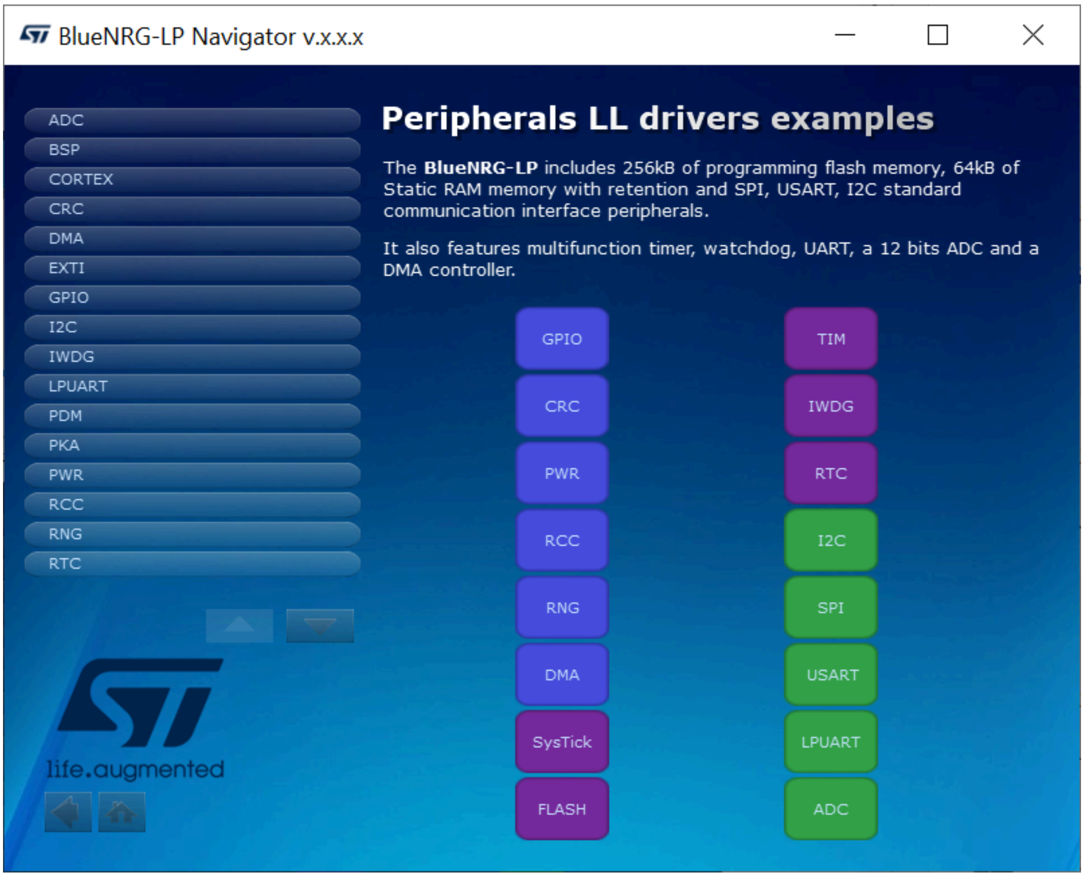
Figure 23. BlueNRG-LP Navigator - BLE demonstration and test applications

3.4 Peripheral driver examples

BlueNRG-LP Navigator includes three sets of peripheral driver examples related to the following categories:

1. LL
2. HAL
3. MIX

Figure 24. BlueNRG-LP Navigator - Peripherals LL driver examples



3.5 2.4 GHz radio proprietary examples

The 2.4 GHz radio proprietary driver provides access to the BlueNRG-LP device radio to send and receive packets without using the Bluetooth link layer.

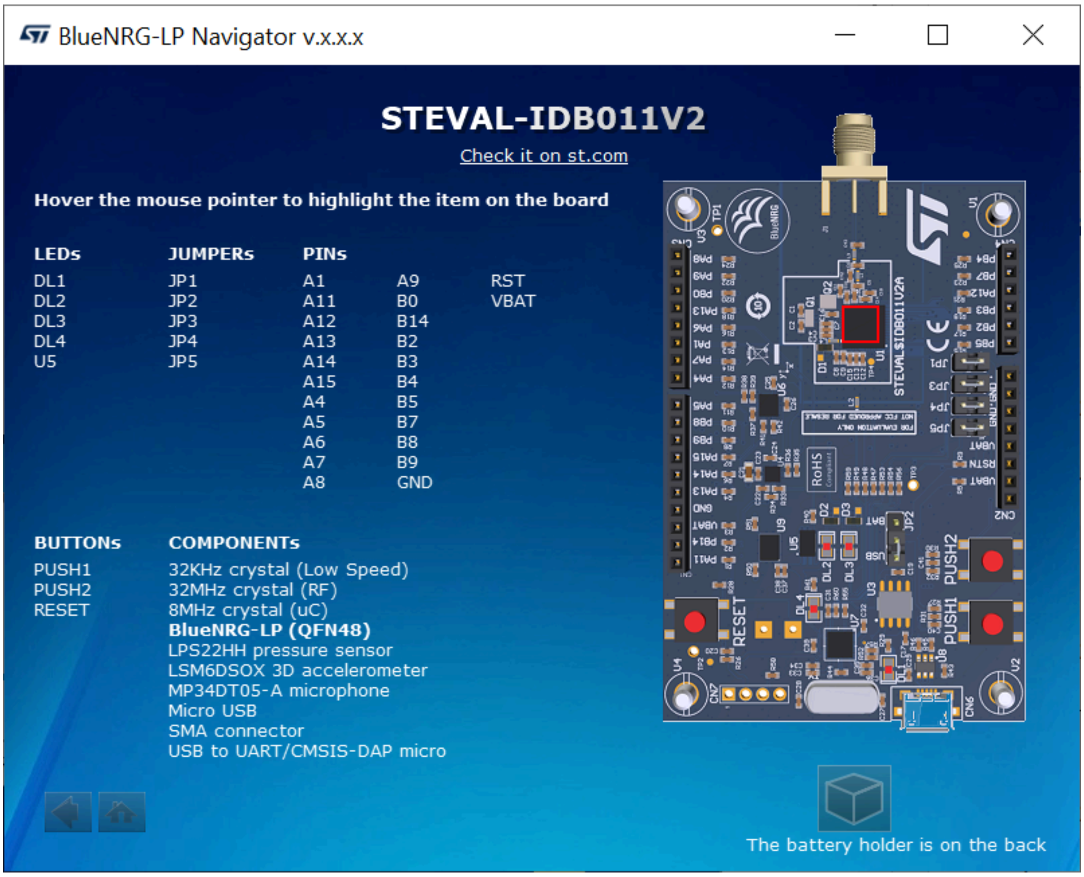
The 2.4 GHz radio proprietary examples are built on top of the 2.4 GHz radio proprietary driver and can be used as reference examples for building other applications which use the BlueNRG-LP Radio.

Figure 25. BlueNRG-LP Navigator - 2.4 GHz radio proprietary examples

3.6 Development kits

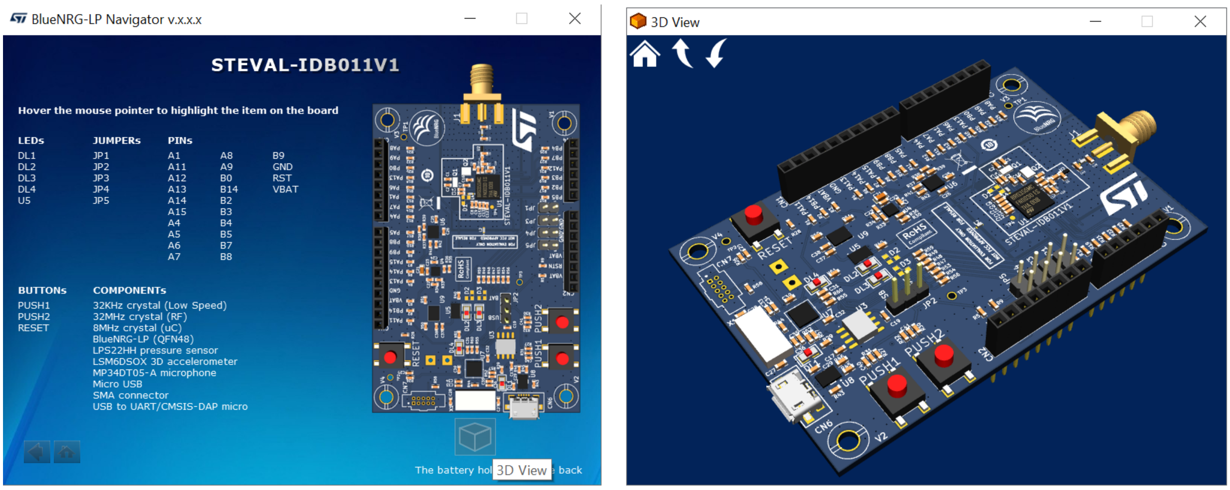
This window displays the available BlueNRG-LP DK kit platforms and corresponding resources. When you hover the mouse pointer over a specific item, the related component is highlighted on the board.

Figure 26. **BlueNRG-LP Navigator - development kit components**



Further, user can get access to the development kit 3D view just selecting the related icon.

Figure 27. **BlueNRG-LP Navigator - development kit 3D view**



3.7 Release Notes and License

The Release Notes and License pages display the DK SW package Release Notes (in html format) and the DK software package license file, respectively.

3.8 Documentation Index

This page displays the DK SW package documentation index providing links to the available documentations (in html and pdf format).

4 BlueNRG-LP/BlueNRG-LPS Radio Init Wizard

The BlueNRG-LP/BlueNRG-LPS Radio Init Wizard is a PC application, which allows defining the proper values required for the correct BlueNRG-LP/BlueNRG-LPS Bluetooth® Low Energy radio initialization, based on the specific user application scenario.

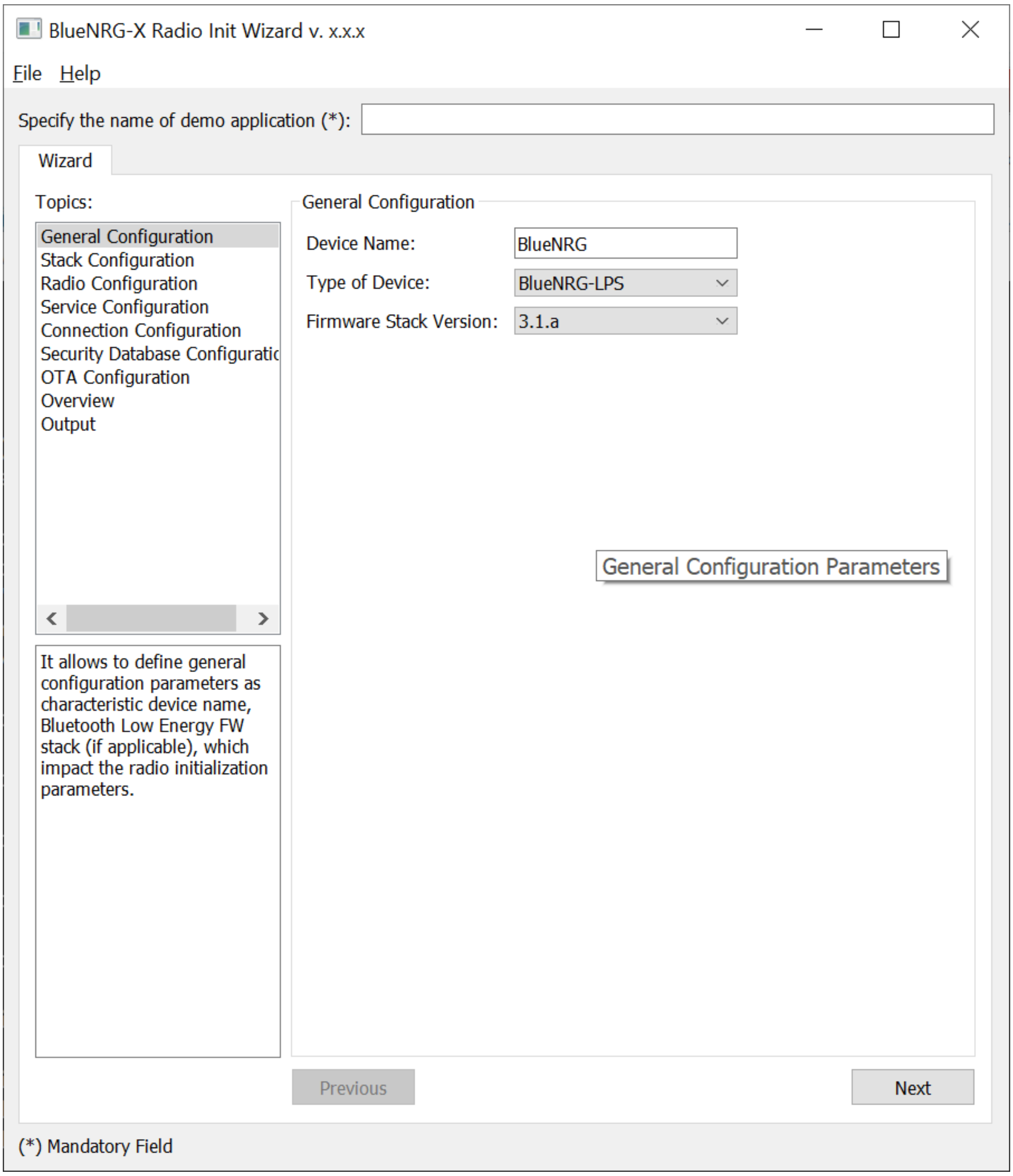
A configuration header file (*_config.h) is generated and can be used on your demonstration application folder.

Note: The BlueNRG-LP/BlueNRG-LPS Radio Init Wizard is provided only for the BlueNRG-LP/BlueNRG-LPS DK SW package (STSW-BNRGLP-DK) supporting the Bluetooth® Low Energy stack v3.x family.

4.1 How to run the Radio Init Wizard

User can run this utility by clicking on the Radio Init Wizard icon under [**Start**]>[**ST BlueNRG-LP_LPS DK X.X.X**].

Figure 28. Radio Init Wizard - general configuration



4.2 Main user interface window

In the left section of the Radio Init Wizard utility, you can select the following topics to define the radio initialization parameters based on the specific Bluetooth® Low Energy application requirements:

1. General configuration
2. Stack configuration
3. Radio configuration
4. Service configuration
5. Connection configuration
6. Security database configuration
7. OTA configuration
8. Overview
9. Output

Note: The Radio Init Wizard tool is used to configure the radio initialization parameter values according to the application needs. It offers an overview of the associated required RAM for the Bluetooth® Low Energy stack. You can also evaluate the impact of each parameter on the overall RAM memory layout to allow tuning the value of each radio initialization parameter (that is, NumOfLinks, ATT_MTU, etc.) according to the available device RAM memory.

Refer to the BlueNRG-LP, BlueNRG-LPS Radio Init Wizard documentation available in the BlueNRG-LP/ BlueNRG-LPS DK SW package (STSW-BNRGLP-DK) for more details about each provided configuration section.

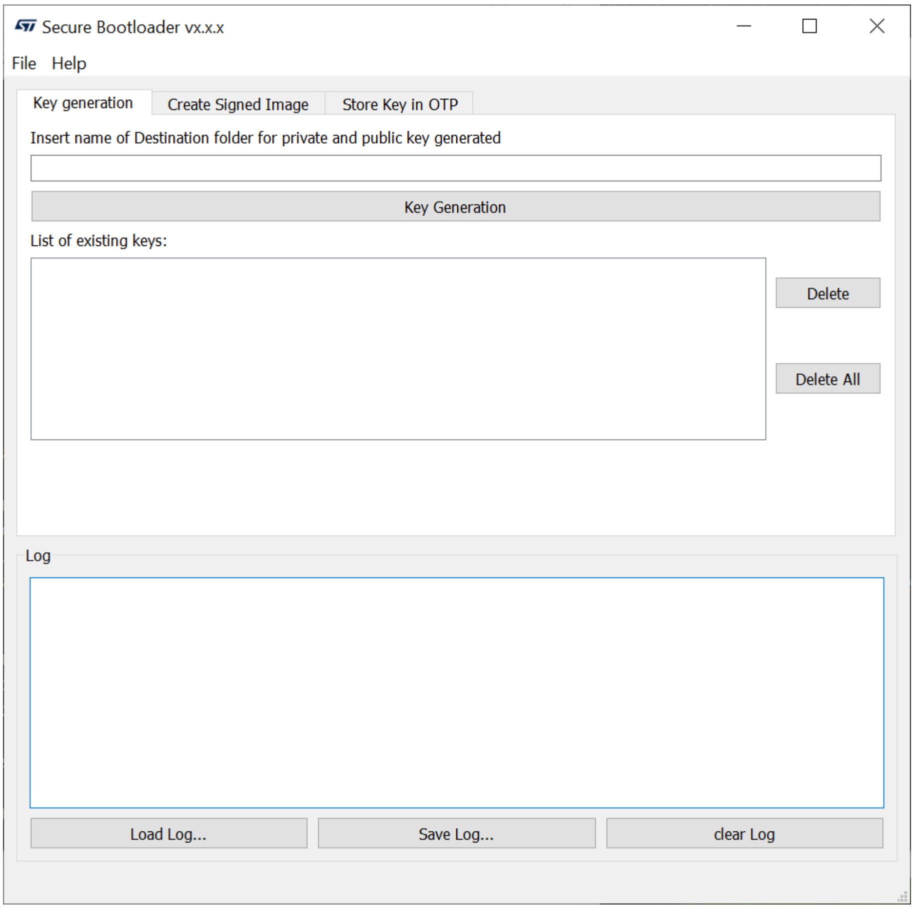
5 Secure Bootloader GUI

The Secure Bootloader GUI PC application exploits the secure bootloader framework functions of the BlueNRG-LP/BlueNRG-LPS device UART bootloader. It allows generating the authentication keys, signing a binary image and activating the secure bootloader through OTP.

This GUI PC application uses the standalone secure bootloader utilities available in the Application/ Secure_bootloader folder.

Important: Once activated on a BlueNRG-LP/BlueNRG-LPS device, the secure bootloader is not reversible.

Figure 29. Secure Bootloader GUI



The application is divided into:

- Key generation
- Created Signed Image
- Store Key in OTP

5.1 Key generation tab

The Key generation tab allows:

- Generating a new key pair (public and private)
- Listing all the existing key pairs on the current database with the option of deleting single keys or deleting them all
- Selecting the destination folder for the generated key pair

Note: The Log tab is also available. It shows the outcomes of the selected operations.

5.2 Create Signed Image tab

The Create Signed Image tab allows:

- Setting the key pair (public and private) to create the file signature
- Selecting the file to be signed
- Setting the name of the signed file
- Creating the signed image

Note: The Log tab is also available. It shows the outcomes of the selected operation.

5.3 Store Key in OTP tab

The Store Key in OTP tab allows:

- Selecting the device COM port
- Selecting the key pair (public and private) to be stored in the device OTP
- Setting the start address of the firmware to be authenticated
- Specifying whether OTP must be locked or not
- Storing the Key in OTP

Note:

- *The Log tab is also available. It shows the outcomes of the selected operations.*
- *The secure boot feature is not reversible and cannot be disabled. When the secure boot feature is activated at reset, the device verifies whether the firmware, in the main Flash, is authenticated or not. The public key stored inside the OTP is used.*
- *As the OTP cannot be changed, it is important to save the keys generated with the utility Key generation tab. If a new firmware update is required, the same keys must be used to generate the signature for the new signed firmware, otherwise the secure bootloader detects the firmware as not authenticated and it cannot be executed.*

6 Programming with the BlueNRG-LP/BlueNRG-LPS system-on-chip

The BlueNRG-LP/BlueNRG-LPS Bluetooth® Low Energy (Bluetooth® Low Energy) stack is provided as a binary library with a set of APIs to control Bluetooth® Low Energy functionality.

Some callbacks are provided for user applications to handle Bluetooth® Low Energy stack events. You have to link the binary library to your application and use the relevant APIs to access the Bluetooth® Low Energy functions and complete the stack event callbacks, managing responses according to application requirements.

A set of software driver APIs is also included to access the BlueNRG-LP/BlueNRG-LPS SoC peripherals and resources (ADC, AES, CRC, DMA, GPIO, I²C, IWDG, LPUART, PWR, RCC, RNG, micro, RTC, SPI, SysTick, TIM, and USART).

The development kit software includes sample code on how to configure BlueNRG-LP/BlueNRG-LPS and use the device peripherals, Bluetooth® Low Energy APIs, and event callbacks.

6.1 Software directories

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

- **Application**: contains BlueNRG-LP/BlueNRG-LPS Navigator, Radio Init Wizard, and Secure Bootloader PC applications
- **Doc**: contains doxygen Bluetooth® Low Energy APIs and events, BlueNRG-LP/BlueNRG-LPS peripheral drivers, Bluetooth® Low Energy demo applications, Peripheral examples, SDK and HAL driver documentation, DK release notes and license file
- **Firmware**: contains prebuilt binary Bluetooth® Low Energy and peripheral driver sample applications
- **Drivers**:
  - **BSP**: SDK drivers providing an API interface to the BlueNRG-LP/BlueNRG-LPS platform hardware resources (LEDs, buttons, sensors, I/O channel)
  - **CMSIS**: BlueNRG-LP/BlueNRG-LPS CMSIS files
  - **External_micro**: drivers framework to support network coprocessor framework with an external microcontroller
  - **Peripherals_Drivers**: BlueNRG-LP/BlueNRG-LPS drivers for device peripherals (ADC, AES, CRC, DMA, clock, GPIO, I²C, IWDG, LPUART, PWR, RCC, RNG, RTC, SPI, SysTick, TIM, and USART).
- **Middlewares**:
  - **Bluetooth® Low Energy**: Bluetooth® Low Energy stack binary library and all the definitions of stack APIs, stack, and events callbacks. Bluetooth® Low Energy stack v3.x configuration header and source files
  - **BLE_Application**: Bluetooth® Low Energy application framework files (Bluetooth® Low Energy stack layers define values, OTA FW upgrade, Bluetooth® Low Energy utilities, master library, GATT, GAP standard profiles, ATT Prepare Write Queue framework)
  - **cryptolib**: AES crypto library
  - **External_micro**: Bluetooth® Low Energy framework to support network coprocessor framework with an external microcontroller
  - **HAL**: hardware abstraction level APIs to abstract certain BlueNRG-LP/BlueNRG-LPS HW/SW features (crash handler, memory utilities, FIFO management, compiler macros, over-the-air utilities for 2.4 GHz radio proprietary solution, general utilities).
  - **NVMDB**: non volatile memory drivers
  - **AESMGR**: aes manager
  - **BLECNTR**: Bluetooth® Low Energy controller manager
  - **PKAMGR**: PKA manager
  - **RNGMGR**: rng manager

- **Projects**
  – **BLE_Examples**: Bluetooth® Low Energy demonstration application including headers, source files and EWARM, Keil® and WiSE-Studio project files
  – **External_micro**: Bluetooth® Low Energy network coprocessor demonstration applications including headers, source files, and EWARM project files
  – **Peripheral_Examples**: with sample applications for the BlueNRG-LP/BlueNRG-LPS peripherals LL, HAL and mixed examples including headers, source files and project files. It also includes the 2.4 GHz radio proprietary examples (in the MIX folder)
- **Utility**: contains the BlueNRG-LP/BlueNRG-LPS patches for IAR, EWARM, and Keil®, MDK-ARM toolchains.

Note: STSW-BNRGLP-DK (BlueNRG-LP/BlueNRG-LPS development kit (DK) software package demonstration applications) targets the BlueNRG-LP BlueNRG-355xy device (flash memory = 256 KB; RAM = 64 KB), provided within the associated evaluation kit.

The demonstration application projects can be easily ported to support the BlueNRG-345xy devices (flash memory = 256 KB; RAM = 32 KB).

For further details on the supported part numbers, refer to the BlueNRG-LP and BlueNRG-LPS datasheets.

Note: The BlueNRG-LP STEVAL-IDB011V1, STEVAL-IDB011V2, STEVAL-IDB010V1 demonstration applications are supported through the STEVAL-IDB011V1 project folder available on each demonstration application EWARM/MDV-ARM/WiSE-Studio folder.

6.2 How to move a demo application from BlueNRG-355xy to BlueNRG-345xy

You can migrate a demonstration application from BlueNRG-355xy (flash memory = 256 KB; RAM = 64 KB) to BlueNRG-345xy (flash memory = 256 KB; RAM = 32 KB) by adding the `CONFIG_DEVICE_BLUENRG_345` as compiler and linker preprocessor options for different toolchains as follows:

- IAR toolchain:
  – in [**Projects**]>[**Options**]>[**General Options**]>[**Target**], select ST BlueNRG-345
  – in [**Projects**][**Options**][**C/C++ Compiler Preprocessor**][**Target**][**Defined Symbols**], add `CONFIG_DEVICE_BLUENRG_345`
  – in [**Projects**]>[**Linker**]>[**Config**]>[**Configuration File symbols definition**], add `CONFIG_DEVICE_BLUENRG_345=1`
- Keil® toolchain:
  – in [**Projects**]>[**Options for Target '...'**]>[**Device**]>[**ST BlueNRG-LP Series**]>[**ST BlueNRG-LP**], select ST BlueNRG-345
  – in [**Projects**]>[**Options for Target '...'**]>[**C/C++**]>[**Preprocessor Symbols**]>[**Define**], add `CONFIG_DEVICE_BLUENRG_345`
  – in [**Projects**]>[**Options for Target '...'**]>[**Linker**]>[**Misc Control**], add `-predefine="-DCONFIG_DEVICE_BLUENRG_345=1"`
- WiSE-Studio toolchain:
  – open the WiSE-Studio.project file and replace BlueNRG-355 with BlueNRG-345
  – in [**Projects**][**Properties**][**C/C++ Build**][**Settings**][**Tool Settings**][**GCC C Compiler**][**Preprocessor**][**Defined Symbols**], add `CONFIG_DEVICE_BLUENRG_345`
  – in [**Projects**][**Properties**][**C/C++ Build**][**Settings**][**Tool Settings**][**GCC Linker**][**Miscellaneous**][**Other linker flags**], add `-Wl,--defsym=CONFIG_DEVICE_BLUENRG_345=1`

Note: *When moving to BlueNRG-LP BlueNRG-345xy, the DTM application must be tuned to fit within the 32 KB RAM memory layout:*

- *select the Bluetooth® Low Energy stack modular option for your application scenario*
- *customize the DTM_Config.h parameters, which affect the RAM memory footprint on the basis of your application scenario (number of links, max ATT_MTU size, number of attributes, etc.)*
- *move DTM_Updater project to BlueNRG-345xy*

For evaluation purposes, a configuration option for the BlueNRG-345xy DTM application is the following one (for the IAR project example below):

- *in [**Projects**]>[**Options**]>[**C/C++ Compiler Preprocessor**]>[**Defined Symbols**], set:* `CONFIG_NUM_MAX_LINKS=3`
- *in [**Projects**]>[**Options**]>[**C/C++ Compiler Preprocessor**]>[**Defined Symbols**], replace* `BLE_STACK_FULL_CONF with BLE_STACK_CUSTOM_CONF`
- *open the header file custom_ble_stack_config.h and set* `#define L2CAP_COS_ENABLED (0U)`

7 BLE Beacon demonstration application

The Bluetooth® Low Energy beacon demo is supported by the BlueNRG-LP development platforms (STEVAL-IDB011V1/STEVAL-IDB011V2/STEVAL-IDB010V1) and the BlueNRG-LPS development platform (STEVAL-IDB012V1, STEVAL-IDB013V1). The demo shows how to configure a device to advertise specific manufacturing data and allow another Bluetooth® Low Energy device to determine whether it is in the Bluetooth® Low Energy beacon device range.

The demo also allows using the extended advertising feature to configure the beacon on a secondary advertising channel.

In the BlueNRG-LPS context, the AoA_Tag project configuration allows configuring a BlueNRG-LPS device as an angle of arrival (AoA) tag for a connectionless scenario.

7.1 BLE Beacon application setup

7.1.1 Initialization

To configure a Bluetooth® LE device to act as a beacon device, you have to initialize correctly the Bluetooth® LE stack as follows:

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

Since the beacon is configuring a device in the advertising non-connectable mode, on the Bluetooth® LE stack v3.1a or later, you can disable the connection support through the `CONNECTION_ENABLED` modular option. As a consequence, using the Bluetooth® LE stack v3.1a or later, you can also initialize the beacon Bluetooth® LE stack as follows:

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

Related links

For Bluetooth® LE stack API documentation, refer to STSW-BNRGLP-DK sw DK package under Docs\BlueNRG-LP_aci_events folder

7.1.2 Manufacturing data

The BLE beacon application advertises the following manufacturing data.

Table 16. BLE beacon application - manufacturing data advertising

Data field	Description	Notes
Company identifier code	SIG company identifier [(1)]	Default is 0x0030 (for STMicroelectronics)
ID	Beacon ID	Fixed value
Location UUID	Beacons UUID	Used to identify specific beacons
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	Complement of the Tx power	Used to establish the distance from the device

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

7.1.3 Non-connectable mode

The Bluetooth® LE beacon device uses the GAP API commands to enter the non-connectable mode.

It sets the advertising configuration parameters as follows:

```
ret = aci_gap_set_advertising_configuration(0,GAP_MODE_GENERAL_DISCOVERABLE,
                                             ADV_PROP_LEGACY,
                                             160, 160,
                                             ADV_CH_ALL,
                                             0,NULL,
                                             ADV_NO_WHITE_LIST_USE,
                                             0, LE_1M_PHY, 0, LE_1M_PHY, 0,0);
```

When using the Bluetooth® LE stack v3.1a or later, with the beacon device is configured in broadcaster mode only, the Bluetooth® LE advertising configuration parameters must be set as follows:

```
ret = aci_gap_set_advertising_configuration(0,GAP_MODE_BROADCAST,
                     ADV_PROP_LEGACY,
                     160, 160,
                     ADV_CH_ALL,
                     0,NULL,
                     ADV_NO_WHITE_LIST_USE,
                     0, LE_1M_PHY, 0, LE_1M_PHY, 0,0);
```

To define the specific selected manufacturer data, the BLE beacon application can use the following GAP API:

```
/* Define the beacon manufacturing payload */
static uint8_t adv_data[] = {
/* Advertising data: Flags AD Type (to be removed if beacon device is configured in broadcast
er only mode) */
0x02,
0x01,
0x06,
/* Advertising data: manufacturer specific data*/
26, //len
AD_TYPE_MANUFACTURER_SPECIFIC_DATA, //manufacturer type
0x30, 0x00, //Company identifier code (Defaultis 0x0030 - STMicroelectronics: To be customize
d
for specific identifier
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, 0x00, // Major number
0x00, 0x07, // Minor number
0xC8 //2's complement of the Tx power (-56dB)};
};
ret = aci_gap_set_advertising_data(0, ADV_COMPLETE_DATA, sizeof(adv_data), adv_data);
```

Note: When using Bluetooth® LE stack v3.1a or later, with the beacon device configured in broadcaster mode only, remove the Flags AD Type bytes 0x02, 0x01, 0x06, since the Flags AD Type is not supported in the broadcaster mode.

The Bluetooth® LE beacon device then enters advertising mode as follows:

```
Advertising_Set_Parameters_t Advertising_Set_Parameters[1];
Advertising_Set_Parameters[0].Advertising_Handle = 0;
Advertising_Set_Parameters[0].Duration = 0;
Advertising_Set_Parameters[0].Max_Extended_Advertising_Events = 0;
/* Enable advertising */
ret = aci_gap_set_advertising_enable (ENABLE, 1, Advertising_Set_Parameters);
```

Note: BLE beacon with Flash Management demonstration application is also available and allows configuring a beacon device. It also shows how to properly handle Flash operations (erase and write) and preserve the Bluetooth® LE radio activities by synchronizing Flash operations with the scheduled Bluetooth® LE radio activities through the `aci_hal_end_of_radio_activity_event()` *event callback timing information.*

7.1.4 Extended advertising mode

A new project configuration (ExtendedAdv) shows how to use the Bluetooth LE specification v5.0 extended advertising feature to configure a beacon also on a secondary advertising channel as follow:

```
#if EXTENDED_ADV
/* Set advertising configuration for extended advertising. */
ret = aci_gap_set_advertising_configuration(1, GAP_MODE_GENERAL_DISCOVERABLE,
                                            ADV_PROP_NONE,
                                            160, 160,
                                            ADV_CH_ALL,
                                            0,NULL, /* No peer address*/
                                            ADV_NO_WHITE_LIST_USE,
                                            0, /* 0 dBm */
                                            (EXT_ADV_PHY==LE_2M_PHY)?LE_1M_PHY:EXT_ADV_PHY,
                                             0, /* 0 skips */
                                             EXT_ADV_PHY, /* Secondary advertising PHY */
                                             1, /* SID */
                                             0 /* No scan request notifications */);
ret = aci_gap_set_advertising_data(1, ADV_COMPLETE_DATA, sizeof(adv_data), adv_data);
#endif
```

The extended advertising is enabled as follows:

```
/* Enable advertising */
#if EXTENDED_ADV
Advertising_Set_Parameters_t Advertising_Set_Parameters[1];
Advertising_Set_Parameters[0].Advertising_Handle = 0;
Advertising_Set_Parameters[0].Duration = 0;
Advertising_Set_Parameters[0].Max_Extended_Advertising_Events = 0;
Advertising_Set_Parameters[1].Advertising_Handle = 1;
Advertising_Set_Parameters[1].Duration = 0;
Advertising_Set_Parameters[1].Max_Extended_Advertising_Events = 0;
ret = aci_gap_set_advertising_enable(ENABLE, 2, Advertising_Set_Parameters);
#endif
```

7.1.5 Periodic advertising mode

A new project configuration (PeriodicAdv) shows how to use the periodic advertising feature related to the Bluetooth® Low Energy specification v5.x to configure a periodic advertising of a beacon payload as follows (refer to the `PERIODIC_ADV` preprocessor option):

```
#ifdef PERIODIC_ADV
  /* Configure periodic advertising */
  ret = aci_gap_set_periodic_advertising_configuration(1,
        PERIODIC_ADV_INTERVAL,
        PERIODIC_ADV_INTERVAL,
        0);

  /* Set periodic advertising data: beacon payload */
  ret = aci_gap_set_periodic_advertising_data(1, 27, adv_data);

  /* Enable periodic advertising */
  ret = aci_gap_set_periodic_advertising_enable(ENABLE, 1)

  printf("Periodic advertising configured\n");

#endif /* PERIODIC_ADV */
```

Note: The periodic advertising requires the extended advertising to be enabled (refer to the `EXTENDED_ADV` preprocessor option).

7.1.6 AoA tag mode

On the BlueNRG-LPS Beacon demonstration application, the AoA_Tag project configuration allows configuring a BlueNRG-LPS device as an angle of arrival (AoA) tag for a connectionless scenario. The AoA tag requires the periodic advertising to be enabled as follows (refer to the `CTE_TAG` and `PERIODIC_ADV` preprocessor options):

```
#ifdef PERIODIC_ADV
  /* Configure periodic advertising */
  ret = aci_gap_set_periodic_advertising_configuration(1,
        PERIODIC_ADV_INTERVAL,
        PERIODIC_ADV_INTERVAL,
        0);

  /* Set periodic advertising data: beacon payload */
  ret = aci_gap_set_periodic_advertising_data(1, 27, adv_data);

#if CTE_TAG /* AoA Tag configuration */
    ret = hci_le_set_connectionless_cte_transmit_parameters(1,
          CTE_LENGTH, CTE_TYPE,
          CTE_COUNT, SWITCHING_PATTERN_LENGTH,
ANTENNA_IDS);

    ret = hci_le_set_connectionless_cte_transmit_enable(1, 1);

  printf("CTE configured\n");
#endif /* CTE_TAG */

  /* Enable periodic advertising */
  ret = aci_gap_set_periodic_advertising_enable(ENABLE, 1)

  printf("Periodic advertising configured\n");

#endif /* PERIODIC_ADV */
```

7.2 BLE Beacon FreeRTOS example

The BLE_Beacon_FreeRTOS example shows how to use FreeRTOS with ST Bluetooth® Low Energy stack v3.x. It configures a BlueNRG-LP/BlueNRG-LPS device in non-connectable advertising mode with specific manufacturing data. The `BTLE_StackTick()` function is called from a FreeRTOS task (BLETask).

A task randomly changes the Minor number in the advertising data, every 500 ms, sending a message via UART each time. Another task sends other messages via UART every 200 ms and generates a short pulse on LED3 (visible with a logic analyzer or oscilloscope).

A low priority has been assigned to the BLETask in this example.

In general, assigning a high priority to the BLETask can give better latency, especially if other tasks occupy a large amount of CPU resources.

Note: *Apart from very short tasks involving sporadic operations while waiting for an event, most other tasks should be assigned a lower priority than the BLETask to avoid slowing Bluetooth® Low Energy operations down.*

8 BLE Serial port demo application

The BlueNRG-LP development platform (STEVAL-IDB011V1/STEVAL-IDB011V2/STEVAL-IDB010V1) and BlueNRG-LPS development platform (STEVAL-IDB012V1/STEVAL-IDB013V1) support the BLE Serial port demo (server and client roles) that implements two-way point to point wireless communication between two Bluetooth® Low Energy devices.

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

- Tx characteristic to enable notifications; before transmitting data, the server sends notifications with the value of the Tx characteristic
- Rx characteristic is a writable characteristic; when the client has to transmit data 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 Serial port service (Bluetooth® Low Energy peripheral device)
- the Client that uses the Serial port service (Bluetooth® Low Energy central device)

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

Table 17. 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 one.

8.1 Peripheral and central device setup

Before establishing the point-to-point wireless Serial port communication between the two Bluetooth LE Serial port devices (server-peripheral and client-central), you need to set the Bluetooth LE device up on both devices by sending a series of API commands to the processor.

8.1.1 Initialization

The Bluetooth LE stack must be correctly initialized before establishing a connection with another Bluetooth LE device through the `aci_gatt_srv_init()` and `aci_gap_init()` APIs.

For `aci_gap_init()`:

- Bluetooth LE Serial port server role

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

- Bluetooth LE Serial port client role

```
aci_gap_init(GAP_CENTRAL_ROLE, 0, 0x08,0,&service_handle,
&dev_name_char_handle, &appearance_c har_handle);
```

Peripheral and central Bluetooth LE roles must be specified in the `aci_gap_init()` command.

Related links

For Bluetooth® LE stack API documentation, refer to STSW-BNRGLP-DK sw DK package under Docs\BlueNRG-LP_aci_events folder

8.1.2 Add service and characteristics

The Serial port service is added to the Bluetooth LE Serial port server device via `aci_gatt_srv_add_service((ble_gatt_srv_def_t *)&serial_port_service);`, where `serial_port_service` is the private service structure allocated for the Serial port service and related characteristics.

The TX characteristic is obtained by using the following command on the Bluetooth LE Serial port server device: `TXCharHandle=aci_gatt_srv_get_char_decl_handle((ble_gatt_chr_def_t *)&serial_port_chars[0]);`, where `serial_port_chars[0]` is the characteristic structure allocated for the TX characteristic (notify property).

The characteristic handle is returned on the `TXCharHandle` variable.

The RX characteristic is obtained by using the following command on the Bluetooth LE Serial port server device: `RXCharHandle = aci_gatt_srv_get_char_decl_handle((ble_gatt_chr_def_t *)&serial_port_chars[1]);`, where `serial_port_chars[1]` is the characteristic structure allocated for the RX characteristic (write property).

The characteristic handle is returned on the `RXCharHandle` variable.

Related links

For Bluetooth® LE stack API documentation, refer to STSW-BNRGLP-DK sw DK package under Docs\BlueNRG-LP_aci_events folder

8.1.3 Entering connectable mode

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

```
/* Configure advertising parameters */
ret = aci_gap_set_advertising_configuration(0,GAP_MODE_GENERAL_DISCOVERABLE,
                                            ADV_PROP_CONNECTABLE|ADV_PROP_SCANNABLE|
                                            ADV_PROP_LEGACY,
                                            ADV_INTERVAL_MIN, ADV_INTERVAL_MAX,
                                            ADV_CH_ALL,
                                            0,NULL,
                                            ADV_NO_WHITE_LIST_USE,
                                            0, LE_1M_PHY, 0,LE_1M_PHY, 0,0);
/* Define advertising data */
static uint8_t adv_data[] = {0x02,AD_TYPE_FLAGS,
FLAG_BIT_LE_GENERAL_DISCOVERABLE_MODE|FLAG_BIT_BR_EDR_NOT_SUPPORTED,
16, AD_TYPE_COMPLETE_LOCAL_NAME,'S','P','o','r','t','_','L','P'};
/* Set advertising data */
ret = aci_gap_set_advertising_data(0,ADV_COMPLETE_DATA, sizeof(adv_data), adv_data);
static Advertising_Set_Parameters_t Advertising_Set_Parameters[1];
Advertising_Set_Parameters[0].Advertising_Handle = 0;
Advertising_Set_Parameters[0].Duration = 0;
Advertising_Set_Parameters[0].Max_Extended_Advertising_Events = 0;
/* Enable advertising */
ret = aci_gap_set_advertising_enable (ENABLE, 1, Advertising_Set_Parameters);
```

8.1.4 Connection with the central device

Once the server device is discoverable by the Bluetooth LE Serial port client device, the client device uses the following APIs to connect to the Bluetooth LE Serial port server device:

```
/* Set scanning configuration parameters */
ret = aci_gap_set_scan_configuration(DUPLICATE_FILTER_ENABLED,SCAN_ACCEPT_ALL,
LE_1M_PHY_BIT, PASSIVE_SCAN, SCAN_INTERVAL, SCAN_WINDOW);
/* Set connection configuration parameters*/
ret = aci_gap_set_connection_configuration(LE_1M_PHY_BIT, CONN_INTERVAL_MIN,
                                           CONN_INTERVAL_MAX, 0, SUPERVISION_TIMEOUT,
                                           CE_LENGTH, CE_LENGTH);
```

To connect with the Bluetooth LE Serial port server device, the following API is used: `ret = aci_gap_create_connection(LE_1M_PHY_BIT, PUBLIC_ADDR bdaddr);`, where `bdaddr` is the peer address of the client device.

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

- on the Bluetooth LE Serial port server device, the typed characters are sent to the Bluetooth LE Serial port client device by notifying the previously added TX characteristic (after notifications are enabled) with: `aci_gatt_srv_notify(connection_handle, TXCharHandle + 1, 0, len, (uint8_t *)(cmd + cmd_buff_start));`
- on the Bluetooth LE Serial port client device, the typed characters are sent to the Bluetooth LE Serial port server device by writing the previously added RX characteristic with: `aci_gatt_clt_write_without_resp(connection_handle, rx_handle + 1 len,(uint8_t *)(cmd+ cmd_buff_start));`, 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 30. Bluetooth LE Serial port client

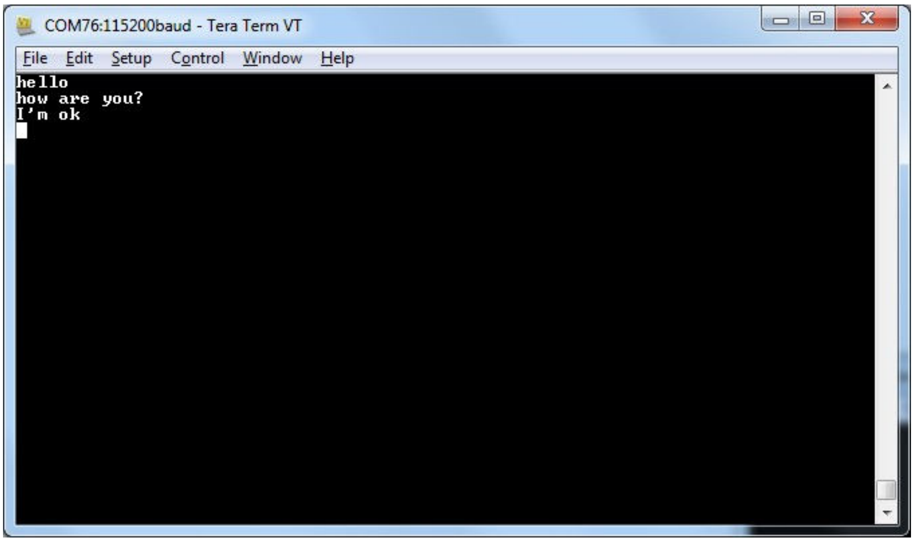
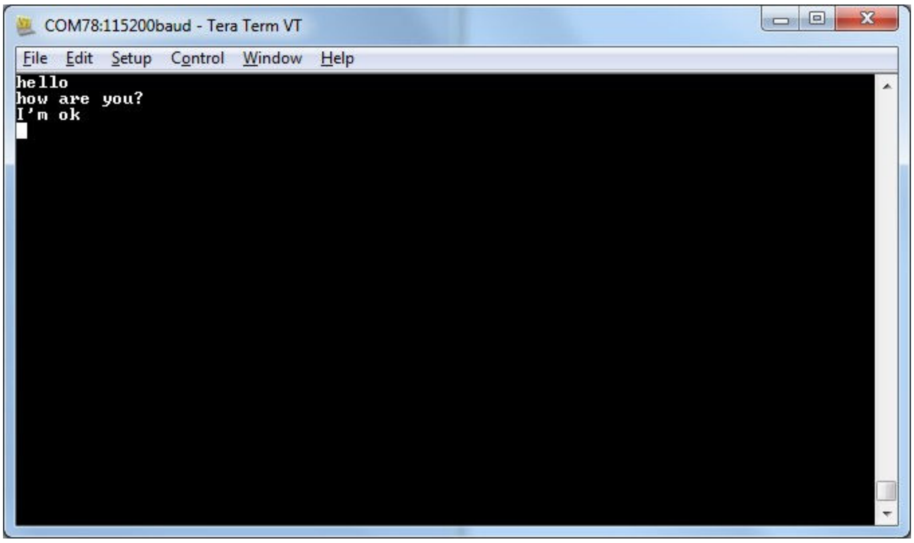


Figure 31. Bluetooth LE Serial port server



Note: The new Client_LongWrites and Server_LongWrites project configurations allow supporting long writes into the characteristic.

9 BLE Serial port master and slave demo application

The BlueNRG-LP development platform (STEVAL-IDB011V1/STEVAL-IDB011V2/STEVAL-IDB010V1) and BlueNRG-LPS development platform (STEVAL-IDB012V1/STEVAL-IDB013V1) support the BLE Serial port master and slave demo that implements point to point wireless communication using a single application, which configures the Serial port client and server roles at runtime.

The Serial port demo application configures a Bluetooth® Low Energy device as central or peripheral using the API: `aci_gap_init(GAP_CENTRAL_ROLE|GAP_PERIPHERAL_ROLE, 0, 0x07, 0, &service_handle,&dev_name_char_h andle, &appearance_char_handle);`

It then initiates a discovery procedure for another Bluetooth® Low Energy device configured with the same Serial port master and slave application image. If the device is found within a random interval, it starts a connection procedure and waits for the connection to be established. If the discovery procedure time expires without finding another Serial port master and slave device, the device enters discovery mode and waits for another Serial port master and slave device to discover and connect to it.

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

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

- Tx characteristic allowing the client to enable notifications; before transmitting data, the server sends notifications with the value of the Tx characteristic.
- Rx characteristic (writable characteristic); when the client has data to be transmitted 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; after the devices have been connected to a PC via USB, you have to open a serial terminal on both with the configuration shown in Table 14.

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 one.

9.1 BLE Serial port master and slave roles

To set up a point-to-point wireless Serial port, two Bluetooth® LE Serial port master and slave devices interact with each other.

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

9.1.1 Initialization

The Bluetooth® LE stack must be correctly initialized before establishing a connection with another Bluetooth LE device via the following commands:

- `aci_gatt_srv_init();`
- `aci_gap_init(GAP_CENTRAL_ROLE|GAP_PERIPHERAL_ROLE, TRUE,0x07, 0, &service_handle, &dev_name_char_handle, &appearance_char_handle);`

The Bluetooth LE peripheral and central roles are specified in the `aci_gap_init()` command.

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 Bluetooth LE serial port master and slave device in discovery mode, a discovery procedure must be started through the following code:

```
/* Set discovery procedure parameters */
ret = aci_gap_set_scan_configuration(DUPLICATE_FILTER_ENABLED,SCAN_ACCEPT_ALL,
        LE_1M_PHY_BIT, PASSIVE_SCAN, DISCOVERY_PROC_SCAN_INT,
      DISCOVERY_PROC_SCAN_WIN);

/* Start general discovery procedure (0x01) */
ret = aci_gap_start_procedure(0x01,LE_1M_PHY_BIT,0,0);
```

9.1.4 Enter connectable mode

The following GAP API commands are used for entering general discoverable mode:

```
/* Configure advertising parameters */
ret = aci_gap_set_advertising_configuration(0,GAP_MODE_GENERAL_DISCOVERABLE,
                                            ADV_PROP_CONNECTABLE|ADV_PROP_SCANNABLE|
                                            ADV_PROP_LEGACY,
                                            ADV_INTERVAL_MIN,
                                            ADV_INTERVAL_MAX,
                                            ADV_CH_ALL,
                                            0,NULL,
                                            0, LE_1M_PHY, 0,LE_1M_PHY, 0,0);
/* Define advertising data */
static uint8_t adv_data[] = {0x02,AD_TYPE_FLAGS,
FLAG_BIT_LE_GENERAL_DISCOVERABLE_MODE|FLAG_BIT_BR_EDR_NOT_SUPPORTED,
13, AD_TYPE_COMPLETE_LOCAL_NAME,'S','P','o','r','t','_','L','P','_','N','e','w'};
/* Set advertising data */
ret = aci_gap_set_advertising_data(0,ADV_COMPLETE_DATA, sizeof(adv_data), adv_data);

/* Enable advertising */
static Advertising_Set_Parameters_t Advertising_Set_Parameters[1];
Advertising_Set_Parameters[0].Advertising_Handle = 0;
Advertising_Set_Parameters[0].Duration = 0;
Advertising_Set_Parameters[0].Max_Extended_Advertising_Events = 0;
ret = aci_gap_set_advertising_enable (ENABLE, 1, Advertising_Set_Parameters);
```

9.1.5 Connection with serial port master and slave client device

In the discovery and mode assignment procedures, the two serial port master and slave applications assume, respectively, client and server roles at runtime. During this initial configuration phase, when a serial port master and slave device is put in discoverable mode and is found by the other serial port 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 commands are used for connecting to the discovered device:

```
* Initialization phase: set connection configuration parameters */
ret = aci_gap_set_connection_configuration(LE_1M_PHY_BIT, 40, 40, 0, 60, 2000, 2000);
/* Create connection */
ret = aci_gap_create_connection(LE_1M_PHY_BIT,
                                discovery.device_found_address_type,
                                discovery.device_found_address);
```

`device_found_address_type` is the address type of the discovered serial port master and slave and `device_found_address` is the peer address of the discovered serial port master and slave device.

Once the two devices are connected, you can set up the 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 Bluetooth LE serial port 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) through the following command:

```
aci_gatt_srv_notify(connection_handle, TXCharHandle + 1, 0, len,
                             (uint8_t *)(cmd + cmd_buff_start))
```

- 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 through the following command:

```
        aci_gatt_clt_write_without_resp(connection_handle, rx_handle + 1,len,
                                                (uint8_t *)(cmd + cmd_buff_start)
```

`connection_handle` is the handle returned upon connection as a parameter of the connection complete an 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 BlueNRG-LP development platform (STEVAL-IDB011V1/STEVAL-IDB011V2/STEVAL-IDB010V1) and BlueNRG-LPS development platform (STEVAL-IDB012V1/STEVAL-IDB013V1) support the BLE remote control application to control a remote device (like an actuator) using the BlueNRG-LP/BlueNRG-LPS device.

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

Table 18. Bluetooth® Low Energy remote advertising data

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

The temperature value is given in tenths of Celsius degrees.

The device is also connectable and exposes a characteristic used to control U5, DL2 and DL3 (on the STEVAL-IDB011V1/STEVAL-IDB011V2), and DL3 and DL4 (on the STEVAL-IDB010V1/STEVAL-IDB012V1/STEVAL-IDB013V1) LEDs on the Bluetooth® Low Energy 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 U5
- bit 1 is the status of DL2
- bit 2 is the status of 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 remains connected to the device indefinitely.

Security is not enabled by default, but can be changed with `ENABLE_SECURITY` (refer to BLE_RC_main.h file). 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 Bluetooth® Low Energy remote control, another Bluetooth® Low Energy device (a BlueNRG-LP, BlueNRG-LPS, or any Bluetooth® Low Energy device) can be used to detect and view broadcast data.

To control one of the LEDs, the device has to connect to a BlueNRG-LP, BlueNRG-LPS Bluetooth® Low Energy remote control device and write in the exposed control point characteristic (service UUID = ed0ef62e-9b0d-11e4-89d3-123b93f75cba, control point characteristic UUID = ed0efb1a-9b0d-11e4-89d3-123b93f75cba).

10.1 BLE remote control application setup

To configure a device to act as a remote control device, follow the procedure described in the following sections.

10.1.1 Initialization

The Bluetooth® LE stack must be correctly initialized before establishing a connection with another Bluetooth LE device through the following commands:

```
aci_gatt_srv_init();
aci_gap_init(GAP_PERIPHERAL_ROLE, 0, 0x07, 0, &service_handle, &dev_name_char_handle,
&appearanc e_char_handle);
```

Related links

For Bluetooth® LE stack API documentation, refer to STSW-BNRGLP-DK sw DK package under Docs\BlueNRG-LP_aci_events folder

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 advertising configuration parameters */
ret = aci_gap_set_advertising_configuration(0,GAP_MODE_GENERAL_DISCOVERABLE,
                                            ADV_PROP_CONNECTABLE|ADV_PROP_SCANNABLE|ADV_PROP_
                                            LEGACY,
                                            (ADV_INTERVAL_MIN_MS*1000)/625,
                                            (ADV_INTERVAL_MAX_MS*1000)/625,
                                             ADV_CH_ALL,
                                             PUBLIC_ADDR,NULL,
                                             ADV_NO_WHITE_LIST_USE,
                                             0, LE_1M_PHY, 0, LE_1M_PHY, 0,0);
/* Define advertising data */
uint8_t adv_data[] = {0x02,AD_TYPE_FLAGS,FLAG_BIT_LE_GENERAL_DISCOVERABLE_MODE|FLAG_BIT_BR_ED
R_NOT_SUPPORTED,0x06,AD_TYPE_MANUFACTURER_SPECIFIC_DATA,0x30,0x00,0x05,0xFF,0xFF};
/* Set advertising data */
ret = aci_gap_set_advertising_data(0, ADV_COMPLETE_DATA, sizeof(adv_data),adv_data);
/* Start advertising */
static Advertising_Set_Parameters_t Advertising_Set_Parameters[1];
Advertising_Set_Parameters[0].Advertising_Handle = 0;
Advertising_Set_Parameters[0].Duration = 0;
Advertising_Set_Parameters[0].Max_Extended_Advertising_Events = 0;
ret = aci_gap_set_advertising_enable(ENABLE, 1, Advertising_Set_Parameters);
```

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 and characteristics is added via:
`aci_gatt_srv_add_service((ble_gatt_srv_def_t *)&rc_service);`, where `rc_service` is the private service structure allocated for the BLE remote service (ed0ef62e-9b0d-11e4-89d3-123b93f75cba) and related characteristics.

The BLE remote control characteristic handle is obtained using the following command:

```
controlPointHandle = aci_gatt_srv_get_char_decl_handle((ble_gatt_chr_def_t*)&rc_chars[0]);
```

The `rc_chars[0]` is the private characteristic structure 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 the read and write characteristic.

10.1.4 Connection with a Bluetooth® Low Energy Central device

When connected to a Bluetooth® Low Energy central device (another BlueNRG-LP/BlueNRG-LPS device or any Bluetooth® Low Energy device), the `controlPointHandle` characteristic is used to control the Bluetooth® Low Energy remote control platform LED.

Each time a write operation is performed on `controlPointHandle`, the `aci_gatt_srv_write_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-LP development platform (STEVAL-IDB011V1/ STEVAL-IDB011V2/STEVAL-IDB010V1) and on the BlueNRG-LPS development platform (STEVAL-IDB012V1/ STEVAL-IDB013V1). It implements a proprietary, Bluetooth low energy (BLE) sensor profile.

This example is useful for building new profiles and applications that use the BlueNRG-LP/BlueNRG-LPS 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.

The acceleration service 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). You can enable or disable the notifications by writing the configuration descriptor of the associated client characteristic.

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 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.

The figure below shows the whole GATT database, including the GATT (0x1801) and GAP (0x1800) services that are automatically added by the stack.

Figure 32. BLE sensor demo GATT database

#	Handle	UUID (16 or 128bit)	Attribute Type	Properties: BRD	RD	WRNO	RESP	WR	NOT	IND	SIGN	wWR	EXT	Initial Parameter Value	Comment
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	E23E78A0CF4A11E18FFC0 002A5D5C51B	**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	340A1B80CF4B11E1AC3600 02A5D5C51B	**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	A32E5520E47711E2A9E300 02A5D5C51B	**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	CD20C480E48B11E2840B00 02A5D5C51B	**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 BLE sensor profile demo: connection with a central device

This section describes how to interact with a central device, while the Bluetooth® LE stack is acting as a peripheral. The central device may be another BlueNRG-LP/BlueNRG-LPS device acting as a Bluetooth® Low Energy master, or any other Bluetooth Low Energy device.

The Sensor device central role demonstration application is able to interact as a Central device with ST Bluetooth LE Sensor Demo. This application searches for ST Bluetooth LE Sensor Demo Peripheral device services and characteristics and gets the related acceleration and temperature sensor values.

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

11.1.1 Initialization

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

```
aci_gatt_srv_init();
aci_gap_init(GAP_PERIPHERAL_ROLE, 0, 0x07, &service_handle, &dev_name_char_handle, &appearanc
e_char_handle);
```

See Bluetooth® LE stack API documentation for more information on these and following commands.

11.1.2 Add service and characteristics

The BlueNRG-LP/BlueNRG-LPS Bluetooth® LE stack can act as server and client. A characteristic is an element in the server database where data is exposed, while a service contains one or more characteristics. The acceleration service and related characteristics are added with the following command:`ret = aci_gatt_srv_add_service(&acc_service);`, where `acc_service` is the allocated structure for the acceleration service.

The free fall and acceleration characteristics can be obtained by:

```
freeFallCharHandle = aci_gatt_srv_get_char_decl_handle(&acc_chars[0]);
accCharHandle = aci_gatt_srv_get_char_decl_handle(&acc_chars[1]);
```

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 related characteristics.

11.1.3 Enter connectable mode

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

```
/* Set advertising configuration parameters */
ret = aci_gap_set_advertising_configuration(0,GAP_MODE_GENERAL_DISCOVERABLE,
                                            ADV_PROP_CONNECTABLE|ADV_PROP_SCANNABLE|ADV_PROP_
                                            LEGACY,
                                            (ADV_INTERVAL_MIN_MS*1000)/625,
                                            (ADV_INTERVAL_MAX_MS*1000)/625,
                                             ADV_CH_ALL,
                                             0,NULL,
                                             ADV_NO_WHITE_LIST_USE,
                                             0, LE_1M_PHY, 0,LE_1M_PHY, 0,0);
/* Define advertsing data */
uint8_t adv_data[] = {0x02,AD_TYPE_FLAGS,
FLAG_BIT_LE_GENERAL_DISCOVERABLE_MODE|FLAG_BIT_BR_EDR_NOT_SUPPORTED,
                              8, AD_TYPE_COMPLETE_LOCAL_NAME,'B','l','u','e','N','R','G'};
/* Set advertising data */
ret = aci_gap_set_advertising_data(0, ADV_COMPLETE_DATA, sizeof(adv_data), adv_data);
/* Enable advertising */
static Advertising_Set_Parameters_t Advertising_Set_Parameters[1];
Advertising_Set_Parameters[0].Advertising_Handle = 0;
Advertising_Set_Parameters[0].Duration = 0;
Advertising_Set_Parameters[0].Max_Extended_Advertising_Events = 0;
ret = aci_gap_set_advertising_enable(ADV_LEGACY, ENABLE, 0, NULL);
```

11.1.4 Connection with a central device

Once the Bluetooth® LE stack is in discoverable mode, it can be detected by a central device.

Any Bluetooth low energy device can connect to the BLE sensor profile demo.

For example, the LightBlue application available in the Apple Store connects the iPhone (versions 4S/5 and above) to the sensor profile device. When you use the LightBlue application, the 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; tapping a service shows the characteristics for that service.

The acceleration characteristic can be notified using the following command: `aci_gatt_srv_notify(connection_handle, accCharHandle + 1, 0, 6, buff);`, where `buff` is a variable containing the three-axis acceleration values.

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

12 BLE Sensor for STBLESensor app

The BLE sensor application for STBLESensor app is supported on the BlueNRG-LP development platform (STEVAL-IDB011V1/STEVAL-IDB011V2/STEVAL-IDB010V1) and on the BlueNRG-LPS development platform (STEVAL-IDB012V1/STEVAL-IDB013V1).

This application shows how to implement a Sensor Demo custom profile application tailored for interacting with the STBLESensor smartphone app.

Once configured and connected, the BlueNRG-LP/BlueNRG-LPS device sends the data collected from the accelerometer sensor and environmental sensor (pressure and temperature) to the STBLESensor smartphone app, which displays this information.

12.1 How to run the STBLESensor app for smartphones

Two versions (Android and iOS) of the smartphone STBLESensor app are available for download.

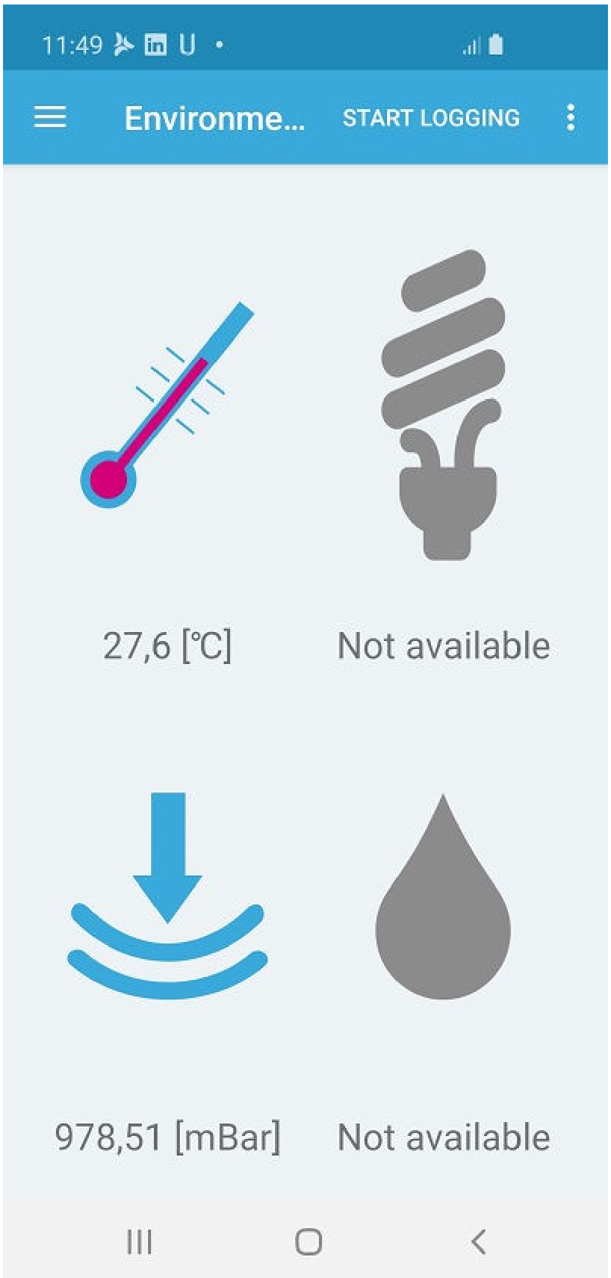
Step 1. Install the app and launch it.

The app starts scanning for the BlueNRG-LP/BlueNRG-LPS Sensor Demo peripheral device. A device called “BlueNRGLP” appears on the screen.

Step 2. Select the “BlueNRGLP” name and connect to the selected platform.

The STBLESensor app enables notifications on the acceleration characteristic and on the environment characteristics (pressure and temperature) and displays the received environment characteristic values on the screen.

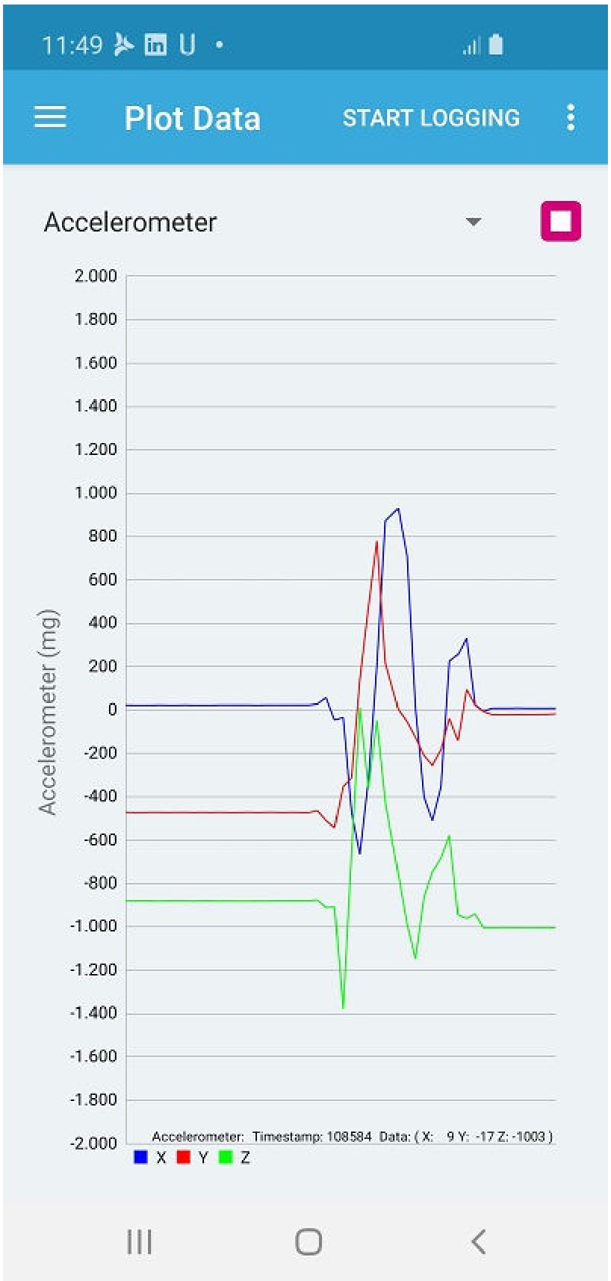
Figure 33. STBLESensor app environment characteristic notifications



Step 3. Select the STBLESensor Demo, Plot Data window and Accelerometer options to plot the received acceleration values.

The STEVAL-IDB011V1/STEVAL-IDB011V2/STEVAL-IDB012V1/STEVAL-IDB013V1/STEVAL-IDB010V1 accelerometer sensor values (X, Y, Z) are displayed on a graphical chart.

Figure 34. STBLESensor app acceleration notification plot



Note: On the BlueNRG-LP platforms (STEVAL-IDB011V1/STEVAL-IDB011V2/STEVAL-IDB010V1), it also supports the BlueVoice library features to perform ADPCM compression and stream voice over Bluetooth LE half-duplex (one-way).

13 BLE sensor profile central demo

The BLE sensor profile central demo is supported on the BlueNRG-LP development platform (STEVAL-IDB011V1/STEVAL-IDB011V2/STEVAL-IDB010V1) and on the BlueNRG-LPS development platform (STEVAL-IDB012V1, STEVAL-IDB013V1). It implements a basic version of the BLE Sensor Profile Central role, which interacts as Central with ST Bluetooth® Low Energy Sensor Demo devices.

This application configures a BlueNRG-LP/BlueNRG-LPS device as a Sensor device Central role, which is able to find, connect and properly configure the free fall, acceleration, and environmental sensor characteristics provided by a Bluetooth® Low Energy development platform configured as a Bluetooth® Low Energy Sensor device Peripheral role (refer to Section 11).

This application uses a new set of APIs to perform the following operations on the BlueNRG-LP/BlueNRG-LPS 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 the available doxygen documentation within the STSW-BNRGLP-DK package. The following master/central binary library is provided in the Library\BLE_Application\Profile_Central\library folder: libmaster_library_bluenrgx.a for IAR, Keil®, and WiSE-Studio in the STSW-BNRGLP-DK package.

14 BLE HID/HOGP demonstration applications

The BlueNRG-LP development platform (STEVAL-IDB011V1/STEVAL-IDB011V2) and the BlueNRG-LPS development platform (STEVAL-IDB012V1/STEVAL-IDB013V1) support the BLE HID/HOGP demonstration applications. They demonstrate a Bluetooth® Low Energy device using the standard HID/HOGP Bluetooth® Low Energy application profile. Keyboard and mouse demo examples are provided.

14.1 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 keyboard device is named ‘STKeyboard’ in the central device list.

To use the HID keyboard, follow the steps below.

Step 1. Insert PIN (123456) to successfully complete the bonding and pairing procedure.

Step 2. Connect the Bluetooth LE development platform to a PC USB port.

Step 3. Open a HyperTerminal window (115200, 8, N,1).

Step 4. Put the cursor focus on the HyperTerminal window.

The keys sent to the central device using the HID/HOGP BLE application profile are also shown on the HyperTerminal window.

14.2 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 left button is the ‘PUSH1’ button and the right button is the ‘PUSH2’ button.

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

Mouse movements are provided by the 3D accelerometer and 3D gyroscope embedded in the BLE development platform.

15 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-LP device on the STEVAL-IDB011V1/STEVAL-IDB011V2/STEVAL-IDB010V1 and STEVAL-IDB012V1, STEVAL-IDB013V1 development platforms.

The Throughput Service contains two characteristics:

- 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 (Bluetooth® Low Energy peripheral device)
2. Client, which uses the service Tx, Rx characteristics (Bluetooth® Low Energy central device).

15.1 BLE throughput setup

To configure two BlueNRG-LP or BlueNRG-LPS platforms with client and server roles, follow the procedure below.

Step 1. Program the client side on one BlueNRG-LP or BlueNRG-LPS platform and reset it.
The platform appears on the screen as a virtual COM port.

Step 2. Open the port in a serial terminal emulator.
The required serial port baudrate is 921600.

Step 3. Program the server side on a second BlueNRG-LP or BlueNRG-LPS platform and reset it.
The platform appears on the screen as a virtual COM port.

Step 4. Open the port in a serial terminal emulator.
The required serial port baudrate is 921600.
The two platforms try to establish a connection. As soon as they get connected, the slave continuously sends notification of a characteristic to the client.

Step 5. Perform an ATT MTU exchange command on the client side to allow the client to increase the ATT_MTU size (247 bytes).

Step 6. Choose the desired link layer packet length: 27 (default), 100 or 251 (maximum allowed) bytes.

Step 7. When bidirectional communication is requested, enable the bidirectional throughput on the client side (the client writes on the RX characteristic).

Step 8. Set the PHY to be used for the communication: 1 Mbps (default), 2 Mbps, etc.

15.2 BLE throughput server commands

After opening the Hyper Terminal (settings = 921600, 8, None,1, None), you can choose and press one of the following interactive options on the server side:

- **u** to send data len update request for 27 bytes
- **m** to send data len update request for 100 bytes
- **U** to send data len update request for 251 bytes
- **l** to enable/disable L2CAP COS txing
- **z** to enable/disable slow throughput
- **1** to change L2C COS MTU value
- **2** to change L2C COS MPS value
- **c** to send connection parameter update request
- **f** to enable/disable flushable PDUs
- **e** to toggle notify
- **p** to print APP flags
- **s** to read LE PHY (TX, RX)
- **d** to set LE RX PHY to Coded
- **D** to set LE TX PHY to Coded
- **t** to set LE TX PHY to 1 Mbps
- **r** to set LE RX PHY to 1 Mpbs
- **T** to set LE TX PHY to 2 Mbps
- **R** to set LE RX PHY to 2 Mpbs
- **x** for system reset
- **?** to print help

15.3 BLE throughput client commands

After opening the Hyper Terminal (settings = 921600, 8, None, 1, None), you can choose and press one of the following interactive options on the client side:

- **u** to send data len update request for 27 bytes
- **m** to send data len update request for 100 bytes
- **U** to send data len update request for 251 bytes
- **a** to send ATT_MTU exchange
- **l** to enable/disable L2CAP COS txing
- **z** to enable/disable slow throughput
- **1** to change L2C COS MTU value
- **2** to change L2C COS MPS value
- **b** to switch bidirectional test on-off
- **n** to send notifications
- **i** to send indication
- **p** to print APP flags
- **s** to read LE PHY (TX, RX)
- **d** to set LE RX PHY to Coded
- **D** to set LE TX PHY to Coded
- **t** to set LE RX PHY to 1 Mbps
- **r** to set LE RX PHY to 1 Mpbs
- **T** to set LE TX PHY to 2 Mbps
- **R** to set LE RX PHY to 2 Mpbs
- **x** for System reset
- **?** to print help

Note: By default, the client and server configurations target a unidirectional throughput test: the server device sends characteristic notifications to the client device. The required serial port baud rate is 921600.

16 BLE notification consumer demonstration application

The BLE ANCS demonstration application configures a BlueNRG-LP/BlueNRG-LPS device as a Bluetooth® Low Energy notification consumer, which facilitates Bluetooth® accessory access to the many notifications generated on a notification provider.

After reset, the demo places the Bluetooth LE device in advertising with device name "ANCSdemo" and sets the device 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 Bluetooth LE device is configured as a notification consumer able to receive the notifications sent from the notification provider.

The BlueNRG-LP development platform (STEVAL-IDB011V1/STEVAL-IDB011V2/STEVAL-IDB010V1) and the BlueNRG-LPS development platform (STEVAL-IDB012V1/STEVAL-IDB013V1) support the BLE notification consumer demonstration application.

17 BLE security demonstration applications

The BlueNRG-LP development platform (STEVAL-IDB011V1/STEVAL-IDB011V2/STEVAL-IDB010V1) and the BlueNRG-LPS development platform (STEVAL-IDB012V1/STEVAL-IDB013V1) support the BLE Security demonstration applications. They provide some basic examples about how to configure two Bluetooth® Low Energy devices as Central and Peripheral, and set up a secure connection by performing a Bluetooth® Low Energy pairing procedure. Once paired, the two devices are also bonded.

The following pairing key generation methods are shown:

- PassKey entry with random pin
- PassKey entry with fixed pin
- Just works
- Numeric Comparison

For each pairing key generation method, a specific project security configuration is provided for both Central and Peripheral devices as shown in the table below. Each Central and Peripheral device must be loaded with the application image targeting the proper security configuration to demonstrate correctly the associated Bluetooth® Low Energy security pairing functionality.

Table 19. BLE security demonstration applications - security configuration 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

17.1 Peripheral device

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

After initialization phase, the 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 Discovery mode with local name `SlaveSec_Ax` (x= 0,1,2,3 depending on the selected security configuration).

Table 20. 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
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 connect.

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

Based on the pairing key generation method, the user may be prompted to perform certain 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 pair and are bonded, the Peripheral device displays the list of its bonded devices and adds the bonded Central device to its white list `aci_gap_configure_white_and_resolving_list()` API.

The Central device starts the service discovery procedure to identify the Peripheral service and characteristics and then enables the TX characteristic notification.

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

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

17.2 Central device

On reset, after initialization, the Central device uses the `Master_SecuritySet()` API for setting the security IO capability and authentication requirements to address the specific selected paring method in combination with the related security settings of the Central device. The Central device application uses the Central/Master library APIs and callbacks to perform the Central device Bluetooth LE operations (device discovery, connection, etc.).

The 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 20. Peripheral device advertising local name parameter value).

When found, the Central connects to the Peripheral device. In order to start the pairing, the Central device waits for the Peripheral device to send a slave security request. Once the security request is received, the Central device starts the pairing procedure. Based on the pairing key generation method, the user may 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 completed, the Central device starts the service discovery procedure to determine the Peripheral TX & RX characteristics.

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

When connected, if the Bluetooth LE platform PUSH1 button is pressed, the Central device disconnects and reconnects with the Peripheral device, which enters undirected connectable mode with advertising filter enabled. Once connected to the Peripheral device, it enters the TX characteristic notification/RX characteristic read cycle again.

Note: *When using a smart phone Central device that implements a random resolvable address, the Peripheral device is not able to accept connection or scan requests from it during the reconnection phase. This is due to the fact that, when disconnecting, the Peripheral device enters the undirected connectable mode with filtering enabled (ADV_WHITE_LIST_FOR_ALL: process scan and connection requests from the White List devices only). Therefore, it is only able to accept the smart phone scan or connection requests if the Privacy Controller is enabled on the Peripheral device.*

A possible simple alternative is to replace the WHITE_LIST_FOR_ALL advertising filter policy on the Peripheral device with NO_WHITE_LIST_USE: the Peripheral device does not enable device filtering after reconnection and is able to accept connection or scan requests from a smart phone by using resolvable random addresses.

18 BLE RC Long Range demonstration application

This application demo shows how to control a remote device (like an actuator) using BlueNRG-LP/BlueNRG-LPS and BLE specification v5.x coded PHY feature.

The demo requires two BlueNRG-LP STEVAL-IDB011V1/STEVAL-IDB011V2/STEVAL-IDB010V1 or two BlueNRG-LPS STEVAL-IDB012V1/STEVAL-IDB013V1 development boards configured, respectively, with client and server configurations.

The boards can be powered by USB cable or AAA batteries (after inserting the batteries, move jumper JP2 to BAT position on the STEVAL-IDB011V1/STEVAL-IDB011V2/STEVAL-IDB010V1 kit platforms). No jumper change is required on the STEVAL-IDB012V1/STEVAL-IDB013V1 kit.

The demonstration application defines two device roles:

- Client device role
- Server device role

18.1 Client and server demo application behavior

The server device starts by entering discoverable mode with a fixed address (0x0280E10000E1). LED2 is blinking as the board is in advertising mode, using 1 Mbps PHY.

The client device (address = 0x0280E10000E0) starts by trying to connect to the server device. LED2 is blinking as the client is trying to connect using 1 Mbps PHY.

Once connected (LED2 always on, on both devices), the client writes into a characteristic to switch LED3 on and off on the server. The state of LED3 changes every 300 ms. It also changes on the client side when the write is sent to the server.

To measure the communication range, you have to place the two boards distant from each other until LED3 blinks on the server at a constant rate. When the LED3 blinking rate is lower, several packets are retransmitted by the client and the boards are at the edge of a stable communication range.

While connected, you have to press the PUSH1 button on one of the boards to change the used PHY to the coded PHY. When the procedure finishes (that is, coded PHY activated), LED1 is switched on.

You can press the PUSH1 button again on the same device to switch PHY back to 1 Mbps.

Note: When PHY is switched by a device, the other device is not able to switch PHY.

You can measure the communication range when using the coded PHY by repeating the test above (placing the boards distant from each other until LED3 blinks on the server at a constant rate). An increased communication range (with respect to 1 Mbps PHY) should be visible.

You can also use the PUSH1 button while not connected to switch advertising or scanning PHY from 1 Mbps to Coded PHY and vice versa. On the client side, some statistics on the link quality are also printed.

19 BLE Controller Privacy demonstration application

This application provides a basic example of Bluetooth low energy controller privacy feature with Bluetooth LE master and slave devices.

19.1 Application scenario

The application scenario is based on two devices, master and slave, configured with `aci_gap_init(privacy flag = 0x02)`, which should perform the following macro steps:

1. Initially, master and slave devices have no info on their security database: the two devices should connect and make a paring and bonding (fixed key: 123456).
2. Once the bonding is completed, the slave calls the `aci_gap_configure_white_and_resolving_list()` API to add its bonded device address to the controller's white list.
3. The master device enables the slave characteristic notification. After the first connection and the pairing/ bonding phase, devices disconnect.
4. The slave enters undirected connectable mode with white list = 0x03 as advertising filter policy (process scan and connection requests only from devices in the White List).
5. The master device performs a direct connection to the detected slave device, which accepts the connection since the master address is on its white list: the two devices reconnect and the slave starts a notification cycle to the master.

Note: When the connection is established, if you press the BLE platform button PUSH1 on one of the two devices, it disconnects and the slave enters the undirected connectable mode with filtering enabled (`WHITE_LIST_FOR_ALL`). This implies that the slave device accepts connection requests only from devices on its white list: the master device is still able to connect to the slave device; any other device connection request is not accepted from the slave device.

20 BLE Multiple Connections demonstration application

This application provides a basic example of a multiple connection scenario.

The Multiple Connections demonstration applications support three device roles:

1. Master_Slave device
2. Master/s device/s
3. Slave/s device/s

20.1 Application roles

The Master_Slave device role allows testing a multiple connection: simultaneous Master and Slave scenario.

It configures the Master_Slave device as central and peripheral device and allows targeting a multiple connection scenario with a MasterSlave device (the collector) which can connect to a given number of peer devices as a master and to a given number of peer devices as a slave.

The Slave device role defines a Peripheral/GATT Server device which connects as slave to the Master_Slave device.

The Master device role defines a Central/GATT Client device which connects as master to the Master_Slave device.

20.2 How to run the application

The MasterSlave device automatically tries to connect to a set of known peer devices (the nodes), to which it is bonded.

Step 1. To bond with a new device, press PUSH1 button on the Slave device and PUSH1 button on the MasterSlave device.

Once the two devices are connected, a bond is created: the MasterSlave tries to connect to the bonded device even after disconnection and the Slave device allows connection only from the bonded MasterSlave.

Step 2. To make the MasterSlave connectable as a slave to a master device (the inquirer), press PUSH2 button on the MasterSlave device.

Step 3. Press PUSH1 on the Master device to detect the MasterSlave device and connect to it.

Note: *In this case, pairing is performed but no bond is created.*

In this scenario, each Slave is a server and periodically sends data to the MasterSlave which acts as server and client. It periodically sends data to the Slaves as a client and to the Masters as a server. Each Master is a client and periodically sends data to the MasterSlave.

Step 4. Open a serial terminal on the associated COM port to show logs from the application.

21 BLE power consumption demo application

The BLE power consumption demo application allows putting the selected BLE device in discovery mode: you can choose from a test menu which advertising interval to use (100 ms or 1000 ms). To measure the BlueNRG-LP/BlueNRG-LPS current consumption, it is necessary to connect a DC power analyzer to the STEVAL-IDB011V1/STEVAL-IDB011V2/STEVAL-IDB010V1 or BlueNRG-LPS STEVAL-IDB012V1/STEVAL-IDB013V1 kit platform power connector (refer to Section 2.18 Current measurements). After that, you can set a connection up with another device configured as a master and measure the related power consumption.

The master role can be covered by another BlueNRG-LP/BlueNRG-LPS kit platform configured with the DTM FW application (DTM_UART_WITH_UPDATER.hex) and running a specific script through the BlueNRG GUI or Script launcher PC applications available in the STSW-BNRGUI SW package.

In the BLE_Power_Consumption demo application project folder, two scripts are provided to configure the master device and create a connection with the BlueNRG-LP/BlueNRG-LPS kit platform under test.

The two scripts allow establishing a connection with 100 ms and 1000 ms as connection intervals, respectively.

The power consumption demo supports the following test commands:

- f: the device is in discoverable mode with a fast interval of 100 ms
- s: the device is in discoverable mode with a slow interval of 1000 ms
- r: to reset the BlueNRG-LP/BlueNRG-LPS
- ?: to display the help menu

22 BLE sync demo application

The BLE sync demo application targets a clock synchronization scenario between master and slave devices.

22.1 Application roles

This demo requires two STEVAL-IDB011V1/STEVAL-IDB011V2/STEVAL-IDB010V1 or two STEVAL-IDB012V1/STEVAL-IDB013V1 development boards configured, respectively, with the Master and Slave project configurations.

The boards can be powered via USB or AAA batteries.

When powering the boards using the batteries, move jumper JP2 to "BAT" position on the STEVAL-IDB011V1/STEVAL-IDB011V2 kit platforms. No jumper change is required on the STEVAL-IDB010V1/STEVAL-IDB012V1/STEVAL-IDB013V1 kit.

22.2 Running the application

Once connected, the master shares its clock information (that is, the network clock) with the slave.

The slave is able to keep the network clock always synchronized with the master while connected, without needing to exchange any other information.

The NETCLOCK library is used to keep the network clock synchronization and implements a function to start a timer that uses the network clock.

This example shows the network clock synchronization by emitting pulses on PA4 at almost the same time (accuracy of ±30 μs). At the same time, the network time is also periodically printed on master and slave and can be read on a serial terminal.

For more accuracy, the `HAL_VTIMER_Tick()` is called inside an interrupt (`CPU_WKUP_IRQHandler()`).

23 BLE power control demo application

The BLE power control demo application targets the following scenarios:

- Dynamic adjustment of TX power level on a connection, based on RSSI
- TX power reporting
- Path loss monitoring

23.1 Application roles

This demo requires two STEVAL-IDB011V1/STEVAL-IDB011V2/STEVAL-IDB010V1/STEVAL-IDB012V1/STEVAL-IDB013V1 boards configured, respectively, with the Peripheral and Central project configurations. Then, the two devices automatically connect.

23.2 Running the application

The Bluetooth stack on each device uses power control procedures to adjust the power of the peer in order to have the signal always in the golden range. The target RSSI and the hysteresis are specified inside the main.h file.

Each time there is a power change on the local or on the remote device, an `hci_le_transmit_power_reporting_event` is received and printed on UART interface.

Path loss monitoring is also enabled: three path loss zones are defined (low, medium and high).

LED1 blinks at different rates in each zone:

- high rate in low path loss zone (boards close to each other)
- medium rate in medium path loss zone
- low rate in high path loss zone (boards far from each other)

When a new zone is entered, the event is printed through UART as well.

24 BLE direction finding demo application

The BLE direction finding application shows how to implement the Bluetooth® Low Energy Direction Finding tag and locator roles using connection CTE mode (angle of departure is selected as CTE type).

24.1 Application roles

This demo requires two BlueNRG-LPS STEVAL-IDB012V1/STEVAL-IDB013V1 boards configured, respectively, with the tag and locator project configurations. Then, the two devices automatically connect.

24.2 Running the application

When connected, the locator device sends constant tone extension (CTE) requests to the peer to receive packets with the CTE field, which is used to collect the IQ samples for direction finding.

The tag device configures the advertising data. When the locator detects and connects to the tag device, the following operations are performed on the tag and locator `hci_le_connection_complete_event()` event callback:

```
/* Tag device */
ret = hci_le_set_connection_cte_transmit_parameters(Connection_Handle,
                                                    CTE_TYPE_SUPPORT,
                                                    TX_SWITCHING_PATTERN_LENGTH,
                                                    antenna_ids);
ret = hci_le_connection_cte_response_enable(Connection_Handle, ENABLE);
/* Locator device */
ret = hci_le_set_connection_cte_receive_parameters(Connection_Handle,
                                                   ENABLE,
                                                   CTE_SLOT_DURATION,
                                                   RX_SWITCHING_PATTERN_LENGTH,
                                                   antenna_ids);
ret = hci_le_connection_cte_request_enable(Connection_Handle,
                                           ENABLE,
                                           CTE_REQ_INTERVAL,
                                           MIN_CTE_LENGTH,
                                            CTE_TYPE);
```

The `hci_le_connection_iq_report_event()` event callback is then triggered. The controller uses this callback to report the IQ samples from the CTE of a received packet.

Note: The actual calculation of the direction from the received IQ samples is beyond the scope of this demonstration application and requires specific algorithms.

25 Direct test mode (DTM) application

Direct test mode (DTM) application allows to configure a BlueNRG-LP/BlueNRG-LPS device as a network coprocessor and target Bluetooth® Low Energy technology evaluation and RF evaluation performances tests, using the BlueNRG GUI tool or other instruments as CBT.

Direct test mode (DTM) is supported by the BlueNRG-LP development platform (STEVAL-IDB011V1/STEVAL-IDB011V2/STEVAL-IDB010V1) and the BlueNRG-LPS development platform (STEVAL-IDB012V1/STEVAL-IDB013V1).

DTM application supports both UART and SPI interfaces.

For more information about the BlueNRG GUI and how to use it with the direct test mode (DTM) application, please refer to “The BlueNRG GUI SW package” user manual (UM2058).

26 BlueNRG-LP/BlueNRG-LPS peripheral driver examples

The BlueNRG-LP/BlueNRG-LPS DK SW package (STSW-BNRGLP-DK) contains a set of peripheral driver examples demonstrating how to use the BlueNRG-LP/BlueNRG-LPS device peripheral drivers (ADC, AES, CRC, DMA,GPIO, I²C,IWDG, LPUART, PWR, RCC, RNG, micro, radio, RTC, SPI, SysTick, TIM, and USART).

These examples are based on the BlueNRG-LP/BlueNRG-LPS LL and HAL drivers available in the STSW-BNRGLP-DK package folder: Drivers\Peripherals_Drivers.

Two peripheral driver frameworks are available:

1. Low level (LL) drivers
2. Hardware abstraction layer (HAL)

The following section provides basic lists of the available BlueNRG-LP/BlueNRG-LPS Peripheral drivers (LL and HAL). For further information on each demonstration application, refer to the related html documentation in the STSW-BNRGLP-DK package in the Docs/BlueNRG-LP_Periph_LL_Examples and Docs/BlueNRG-LP_Periph_HAL_Examples folders.

Note: Specific examples for some IPs and device features (2.4 GHz radio proprietary solution, power save Modes, etc.) are provided in the Projects\Peripheral_Examples\MIX folder.

26.1 ADC examples

ADC battery sensor (LL/HAL) This example shows how to sample the supplied voltage using the ADC peripheral.

ADC temperature sensor (LL) This example shows how to sample the temperature using the internal temperature sensor.

ADC conversion sequence (LL/HAL) This example shows how to use the sequencer of the ADC peripheral to acquire samples from different sources.

ADC analog microphone (LL) This example shows how to use the ADC peripheral to interface an analog microphone.

ADC DMA (HAL) This example shows how to use an ADC peripheral to perform a continuous ADC conversion from the ADC pin PB3 through DMA.

ADC Watchdog (LL) This example shows how to use the ADC watchdog to trigger a watchdog event if a signal goes outside the defined low and high thresholds.

26.2 BSP examples

BSP COM (LL) This example shows basic UART communication.

BSP LEDs, Buttons (LL) This example shows how to configure the EXTI and use GPIOs to toggle the user LEDs available on the board when a user button is pressed.

BSP Sensors (LL) This example shows how to configure the inertial and the pressure sensors.

26.3 CRC examples

CRC calculate check (LL) This example shows how to configure the CRC calculation unit to compute a CRC code for a given data buffer.

CRC example (LL/HAL) This example shows how to configure the CRC calculation unit to compute a CRC code for a given data buffer, based on a fixed generator polynomial (default value 0x4C11DB7).

CRC User Defined Polynomial (LL/HAL) These examples show how to configure the CRC calculation based on a user-defined generating polynomial.

CRC Polynomial Update (MIX) This example shows how to use the CRC peripheral through the BLUENRG_LP CRC HAL and LL API (used for performance improvement).

26.4 DMA examples

DMA mem to mem (LL)	This example shows how to use a DMA channel to transfer a word data buffer from embedded SRAM to embedded SRAM. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).
DMA RAM to RAM (MIX)	This example shows how to use a DMA to transfer a word data buffer from RAM to embedded SRAM through the BLUENRG_LP DMA HAL and LL API (used for performance improvement).

26.5 EXTI example

EXTI Toggle LED ON with IT (LL)	This example shows how to configure the EXTI and use GPIOs to toggle the user LEDs.

26.6 Flash example

Erase Program (HAL)	This example shows how to use the basic Flash operations such as erase/write/read.
Write Protection (HAL)	This example demonstrates how to use the basic Flash operations such as erase/write/read.

26.7 GPIO examples

GPIO Infinite LED Toggling (LL)	This example shows how to configure and use GPIOs to toggle the on-board user LEDs every 250 ms.
GPIO IO toggle (HAL)	This example shows how to configure and use GPIOs through the HAL API.
GPIO EXTI (HAL)	This example shows how to configure external interrupt lines.

26.8 HAL examples

HAL time base (HAL)	This example shows how to customize HAL using a general-purpose timer as main source of time base, in place of Systick.
HAL RTC Alarm (HAL)	This example shows how to customize HAL using RTC alarm as main source of time base, in place of Systick.
HAL TimeBase RTC WKUP (HAL)	This example shows how to customize HAL using RTC wakeup as main source of time base, in place of Systick.

26.9 I²C examples

I2C_IT_Pressure (LL)	This example shows how to handle the reception of one data byte from the on-board pressure sensor to the device via I²C in the interrupt mode.
I2C DMA (LL/HAL)	This example shows how to handle I²C data buffer transmission/reception between two boards via DMA.
I2C Polling (LL/HAL)	This example shows how to handle I²C data buffer transmission/reception between two boards in polling mode.
I2C Restart Adv IT (HAL)	This example shows how to perform multiple I²C data buffer transmission/reception between two boards in interrupt mode and with restart condition.
I2C Restart IT (LL/HAL)	This example shows how to handle single I²C data buffer transmission/reception between two boards in interrupt mode and with restart condition.
I2C AdV IT (HAL)	This example shows how to handle I2C data buffer transmission/reception between two boards, using an interrupt starting a new communication transfer after the first.
I2C IT (HAL)	This example shows how to handle I2C data buffer transmission/reception between two boards, using an interrupt.

26.10 I²S examples

I2S audio demo (HAL)	This example plays a pre-recorded audio frame, saved in the MCU Flash memory, via the STA350BW device soldered on the X-NUCLEO-CCA01M1 expansion board connected to a BlueNRG-LP evaluation kit.
I2S_Receiver (HAL)	This application is designed to configure the I²S peripheral on the master board to receive data from the slave board.
I2S_Transmitter (HAL)	This application is designed to configure the I²S peripheral on the slave board to transmit data to the master board.

26.11 IWDG examples

IWDG Refresh until User Event Init (LL)	This example shows how to configure the IWDG peripheral to ensure periodical counter update and generate an MCU IWDG reset when PUSH1 button is pressed.
IWDG Window Mode (HAL)	This example shows how to periodically update the IWDG reload counter and simulate a software fault that generates an MCU IWDG reset after a preset amount of time.
IWDG Reset (HAL)	This example shows how to handle the IWDG reload counter and simulate a software fault that generates an IWDG reset after a preset amount of time.

26.12 LPUART example

LPUART TX, RX (LL)	This example shows how to configure GPIO and LPUART peripherals to receive characters on LPUART RX. This example is based on the LPUART LL API. The peripheral is initialized using LL unitary service functions for optimization purposes (performance and size).

26.13 Micro MIX examples

Hello world	This is an example for the basic 'BlueNRG-LP/BlueNRG-LPS Hello World' application with crash info handling and device cut information. When you connect the BlueNRG-LP/BlueNRG-LPS platform to a PC USB port and open a specific PC tool/program (like Tera Term), the "Hello World: BlueNRG-LP/BlueNRG-LPS (x.x) is here!" message is displayed.
Power Save Modes test	This test provides an example for the following BlueNRG-LP/BlueNRG-LPS power save modes:

POWER_SAVE_MODE_SHUTDOWN	It turns the BlueNRG-LP/BlueNRG-LPS device off. The only wakeup source is a low pulse on the RSTN pad (there is no smart power management as memory retention is not part of this power save level configuration).
POWER_SAVE_LEVEL_STOP_WITH_TIMER	It puts the BlueNRG-LP/BlueNRG-LPS in deep sleep and the timer clock sources (LSI or LSE) remain running. Wakeup is possible from GPIOs (PA8 to PA11 and PB0 to PB7), RTC, IWDG, BlueCore and the HAL Virtual Timers.
POWER_SAVE_LEVEL_NOTIMER	It puts the BlueNRG-LP/BlueNRG-LPS in deep sleep. All the peripherals and clock sources are turned off. Wakeup is possible only from GPIOs (PA8 to PA11 and PB0 to PB7).
POWER_SAVE_LEVEL_CPU_HALT	It puts the BlueNRG-LP/BlueNRG-LPS in CPU halt. Only the CPU is halted. The rest of the chip continues running normally. The chip will wake up from any interrupt.

26.14 PDM demonstration application

PDM_DigitalMicrophone (LL)	This example shows how to use the PDM port to interface a digital MEMS microphone.

26.15 Public Key Accelerator (PKA) demonstration application

PKA ECC Sign (LL)

The BlueNRG-LP/BlueNRG-LPS PKA demonstration application is supported by the BlueNRG-LP/BlueNRG-LPS development platforms. It provides a basic example on how to use the available PKA driver APIs to perform a basic PKA processing and check the results.

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

Note: This peripheral is used by the BlueNRG-LP/BlueNRG-LPS Bluetooth low energy stack during the security pairing procedure, so the user application must not use it in the meantime.

The PKA demonstration application performs the following steps:

1. Starting from the PKA known point on the `PKS_SetData()` ellipse with `PKA_DATA_PCX`, `PKA_DATA_PCY` and from a random generated key A, it performs a PKA process which generates a new point A on the ellipse.
2. The same process is repeated from a new generated random key B, leading to a new point B on the ellipse.
3. A new PKA process starts using the key A with the point B coordinates generating a new point C which is still on the same ellipse.

26.16 PWR examples

PVD (LL)

This example shows how the PVD measures a voltage below the comparator and how an interrupt is raised to the CPU in the SYSCFG block.

26.17 2.4 GHz radio proprietary MIX examples

The 2.4 GHz radio proprietary low level driver provides access to the BlueNRG-LP/BlueNRG-LPS device 2.4 GHz radio to send and receive packets without using the Bluetooth link layer.

The available 2.4 GHz radio proprietary examples are:

AutomaticChMgm TX example where the `INC_CHAN` Action Tag is used to automatically change the channel.

Beep TX example where the device continuously sends a packet to three different channels.

BeepMultiState TX example with multi-state functionality.

Serial port Point-to-point communication generating a two-way Serial port (encryption configuration is also provided).

RemoteControl A basic remote control scenario by pressing the PUSH1 button on the device makes toggle the LED1 on the receiver device.

Sleep It demonstrates point-to-point communication with sleep management.

Sniffer A sniffer application in a selected channel and a defined Network ID.

SnifferMultiState A sniffer application with multi-state functionality.

StarNetwork A star network example where a Master asks for packets to the slaves of the network.

Tx Rx Point-to-point communication with computation of packet error rate (PER).

Throughput TX, RX Throughput test example (unidirectional with one TX and one RX device, and bidirectional with two TX devices and one RX device)

OTA Client, Server OTA firmware upgrade framework example based on 2.4 GHz radio proprietary low-level driver.

26.18 RCC examples

RCC HSE Startup test (MIX) It shows how to perform HSE startup time measurement.

26.19 RNG examples

RNG Generate Random Numbers (LL) This example shows how to configure RNG to generate 32-bit long random numbers. The peripheral initialization uses LL unitary service functions for optimization purposes (performance and size).

RNG Multi RNG (HAL) This example shows how to configure the RNG using the HAL API. This example uses the RNG to generate 32-bit long random numbers.

26.20 RTC examples

RTC wakeup (LL) This example shows how to configure GPIO, USART and RTC peripherals to set the power save level stop with timer.

RTC Alarm (LL/HAL) Configuration of the RTC LL API to configure and generate an alarm using the RTC peripheral. In this example, the Time is set to 23:59:50 and the Alarm must be generated on 00:00:00.

RTC Autocalibration (HAL) This example shows use of the LSE clock source auto-calibration to get a precise RTC clock.

RTC Calendar (LL/HAL) This example shows the configuration of the calendar.

RTC Exit Standby with Wake-Up Timer (LL) This example shows configuration of the RTC to wake up from Standby mode using the RTC Wakeup timer.

RTC Wakeup Calendar (LL) This example shows configuration of the LL API to set the RTC calendar. When a reset occurs the RTC configuration is not lost.

26.21 SPI examples

SPI Half Duplex IT (LL) This example shows how to configure GPIO and SPI peripherals to transmit bytes from an SPI Master device to an SPI Slave device in Interrupt mode.

SPI Half Duplex DMA (LL) This example shows how to configure GPIO and SPI peripherals to transmit bytes from an SPI Master device to an SPI Slave device in DMA mode.

SPI Full Duplex IT Master/ Slave (LL/HAL) These examples show how to perform data buffer transmission and reception between two boards via SPI using Interrupt mode.

SPI DMA Master/Slave (LL/ HAL) These examples show how to perform data buffer transmission and reception between two boards via SPI using DMA.

SPI Full Duplex Polling Master/ Slave (HAL/MIX) These examples show how to perform data buffer transmission and reception between two boards via SPI using Polling mode.

SPI Half Duplex Polling IT Master/Slave (MIX) These examples show data buffer transmission/reception between two boards via SPI using Polling (LL driver) and Interrupt modes (HAL driver).

SPI Full Duplex IT Master (HAL) This example shows how to perform the data buffer transmission and reception between two boards via SPI using the interrupt mode (CPOL = 1 ; CPHA =0 ; NSSP =1).

SPI Full Duplex IT Slave (HAL) This example shows how to perform the data buffer transmission and reception between two boards via SPI using the in nterrupt mode (CPOL = 1 ; CPHA =0 ; NSSP =0]

.

26.22 TIM examples

TIM Break and Dead time (LL) This example shows how to configure the TIM peripheral to generate three center-aligned PWM and complementary PWM signals, insert a defined dead time value, use the break feature and lock the break and dead-time configuration.

TIM_Frequency_Divider (LL/HAL) The TIM1 peripheral is configured to generate an output signal with a frequency equal to the half of the input signal acquired.

TIM Input capture (LL/HAL/ MIX) This example shows how to use the TIM peripheral to measure a periodic signal frequency provided by an external signal generator.

TIM Output Compare Init (LL) This example shows how to configure the TIM peripheral to generate an output waveform in different output compare modes.

TIM time base (LL)	This example shows how to configure the TIM peripheral to generate a time base.
TIM OC active (HAL)	This example shows how to use the TIM peripheral in Output Compare Active mode (when the counter matches the capture/compare register, the corresponding output pin is set to its active state).
TIM OC inactive (HAL)	This example shows how to use the TIM peripheral in Output Compare Inactive mode with the corresponding Interrupt requests for each channel.
TIM OC toggle (HAL)	This example shows how to use the TIM peripheral to generate four different signals at four different frequencies.
TIM PWM output (LL/HAL)	This example shows how to use the TIM peripheral in PWM (pulse width modulation) mode.
TIM PWM Train (LL)	Configuration of a timer to generate a fixed number of pulses in Output Compare mode.
TIM time base (HAL)	This example shows how to use the TIM peripheral to generate a time base of one second with the corresponding interrupt request.
TIM One Pulse (LL/HAL)	This example shows how to use the TIM peripheral to generate a single pulse when an external signal rising edge is received on the timer input pin.
TIM One Pulse TI2 Trigger (LL)	Configuration of a timer to generate a positive pulse in Output Compare mode with a length of tPULSE and after a delay of tDELAY.
TIM PWM Input (HAL)	This example shows how to use the TIM peripheral to measure the frequency and duty cycle of an external signal.

26.23 UART examples

USART TxRx DMA (LL)	This example shows how to configure USART peripheral to send characters asynchronously to/from an HyperTerminal (PC) in DMA mode.
USART Rx Cont IT (LL)	This example shows how to configure GPIO and USART peripheral to continuously receive characters from Hyper Terminal in Asynchronous mode using Interrupt mode.
USART Tx (LL)	This example shows how to configure GPIO and USART peripherals to send characters asynchronously to an Hyper Terminal in Polling mode. If the transfer cannot be completed within the allocated time, a timeout allows exiting from the sequence with a Timeout error code.
USART Tx IT (LL)	This example shows how to configure GPIO and USART peripheral to send characters asynchronously to Hyper Terminal in Interrupt mode.
USART_SPI_Master (LL)	This example shows how to configure the USART peripheral to send characters synchronously to/from the LL SPI_IT_Slave_Init example in the DMA mode.
USART_SPI_Slave (LL)	This example shows how to configure the USART peripheral to send characters synchronously to/from the LL SPI_IT_Master_Init example in DMA mode.
USART_TxRx_Inverted (LL)	This example shows how to configure the GPIO and USART peripherals to send characters asynchronously in the DMA mode with the Tx Rx pin level inverted.
UART DMA (HAL)	This example shows how to perform UART transmission (transmit/receive) in DMA mode between a board and a Hyper Terminal application.
UART IT (HAL/MIX)	This example shows how to perform UART transmission (transmit/receive) in Interrupt mode between a board and a Hyper Terminal application.
UART printf (HAL)	This example shows how to perform re-routing of the C library printf function to the UART. The UART outputs a message on the Hyper Terminal.
USART HyperTerminal TxPolling RxIT (MIX)	Use of a UART to transmit data (transmit/receive) between a board and the Hyper Terminal application both in Polling and Interrupt modes.
USART Tx Rx Hw Flow Control IT (LL/HAL)	This example shows how to configure GPIO and USART peripheral to send characters asynchronously to/from Hyper Terminal with hardware flow control mode in Interrupt mode.
UART_CircularDMA (HAL)	This example shows how to use the HAL UART API for the reception in the circular DMA mode.

27 STEVAL-IDB011V1 schematic diagrams

Figure 35. STEVAL-IDB011V1 circuit schematic (1 of 3)

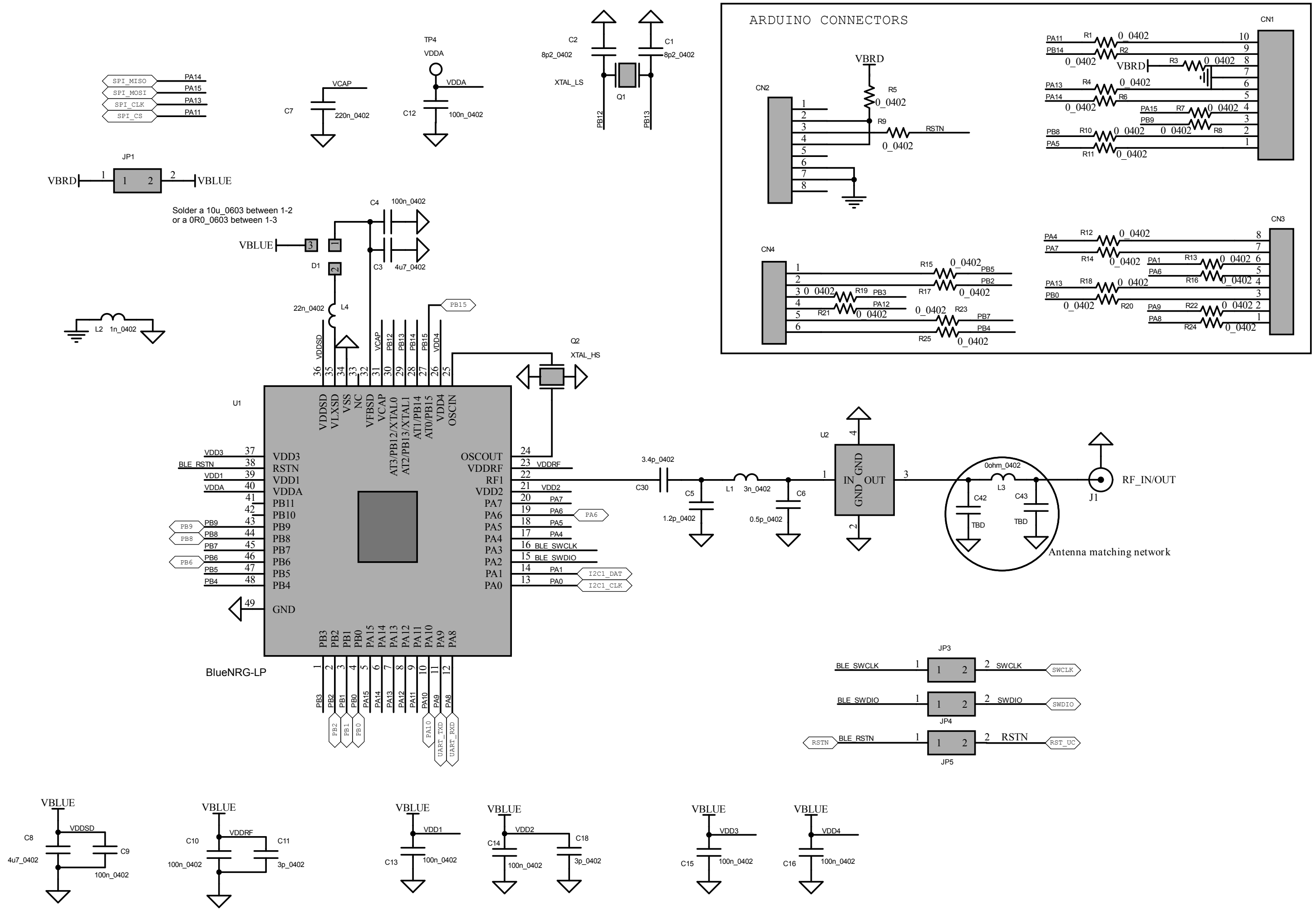


Figure 36. STEVAL-IDB011V1 circuit schematic (2 of 3)

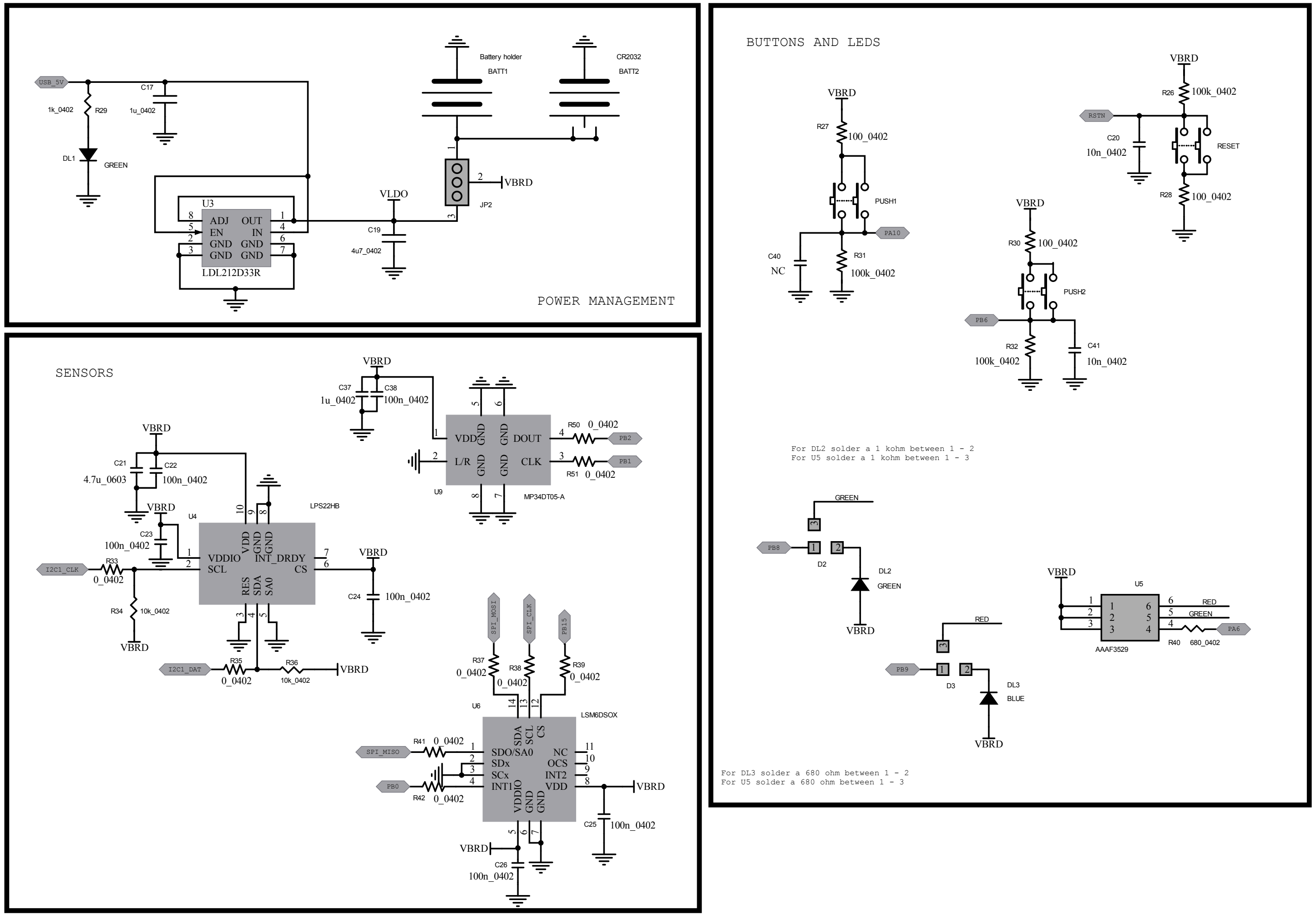
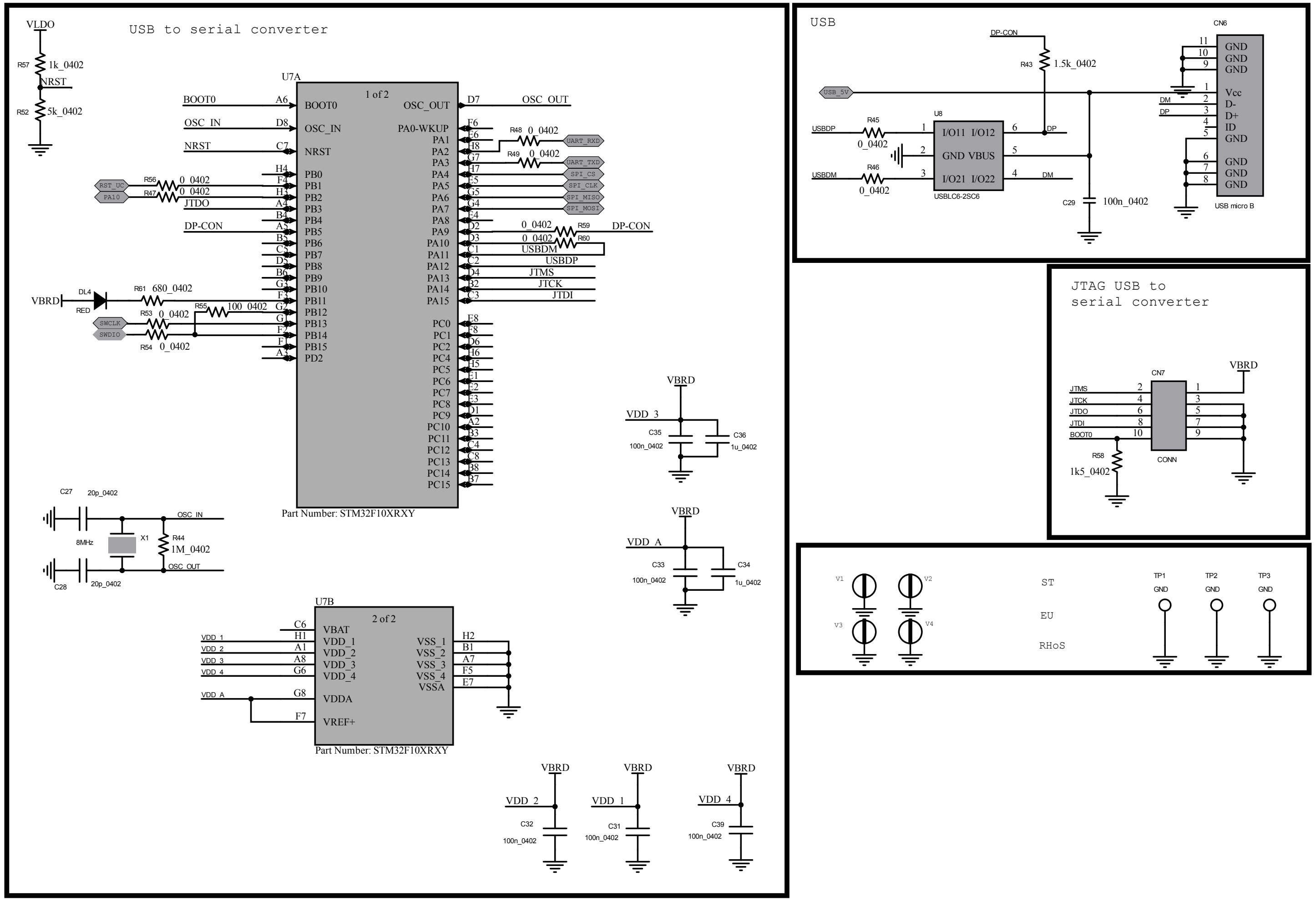


Figure 37. **STEVAL-IDB011V1 circuit schematic (3 of 3)**



28 STEVAL-IDB011V2 schematic diagrams

Figure 38. STEVAL-IDB011V2 circuit schematic (1 of 3)

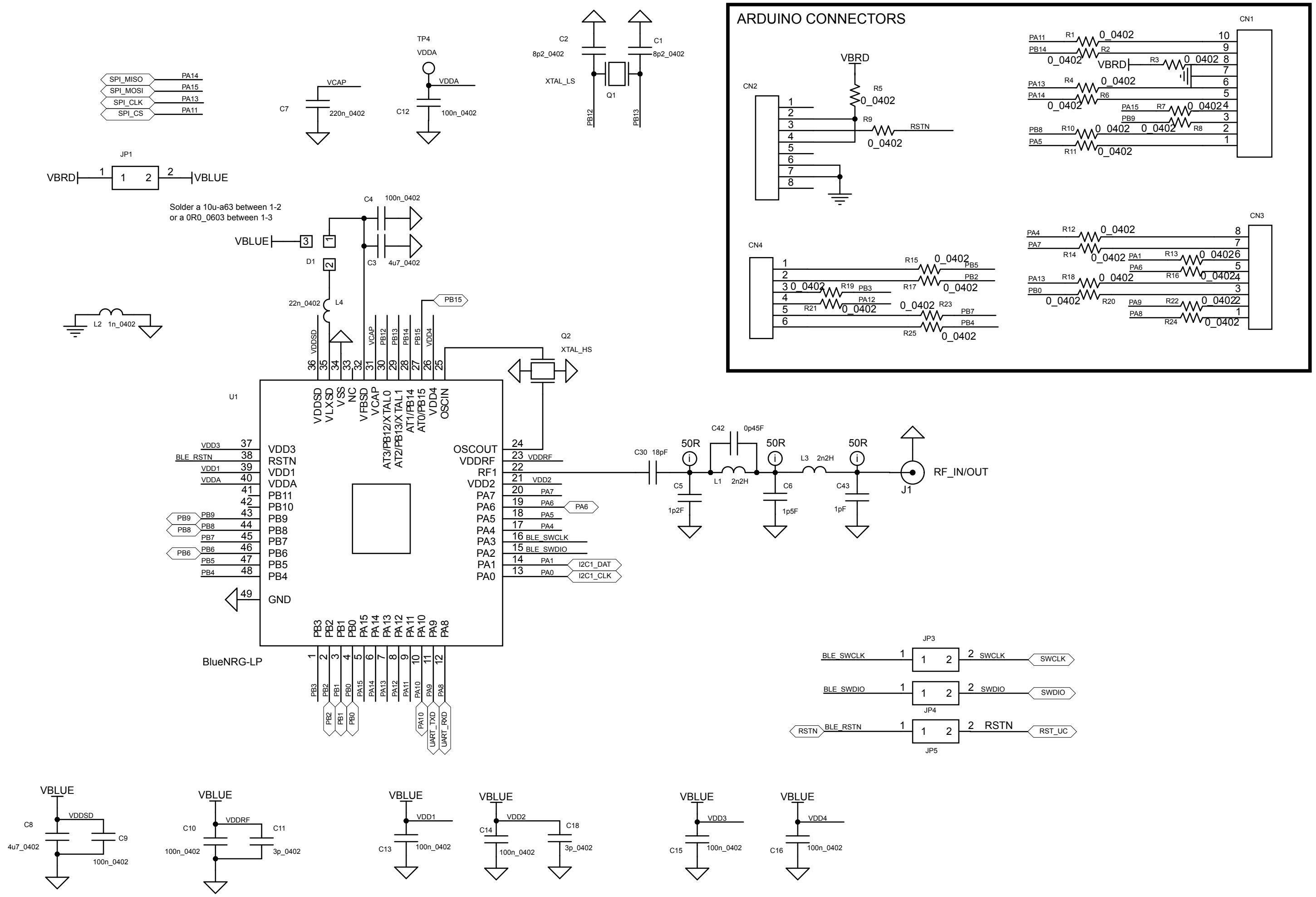


Figure 39. STEVAL-IDB011V2 circuit schematic (2 of 3)

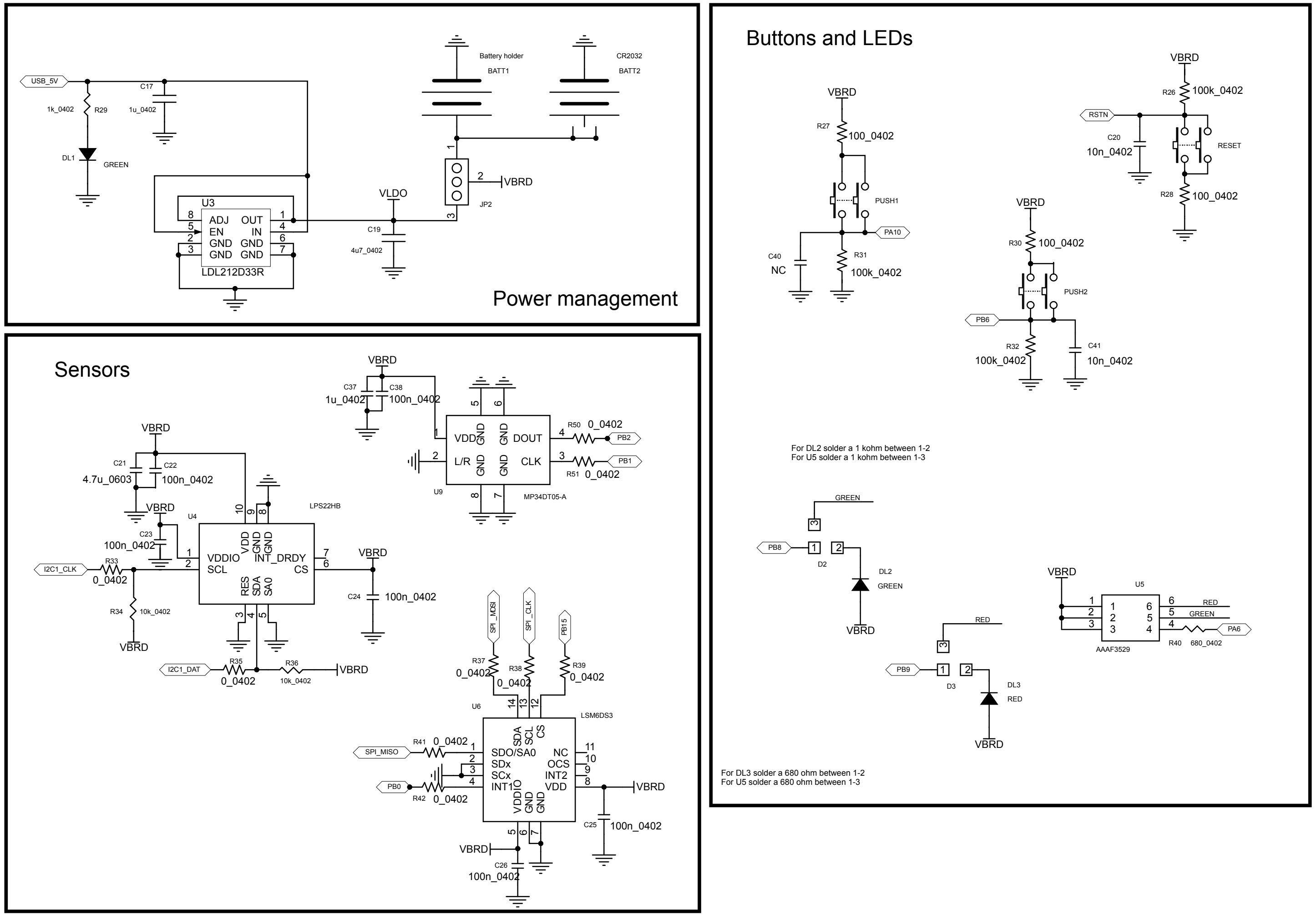
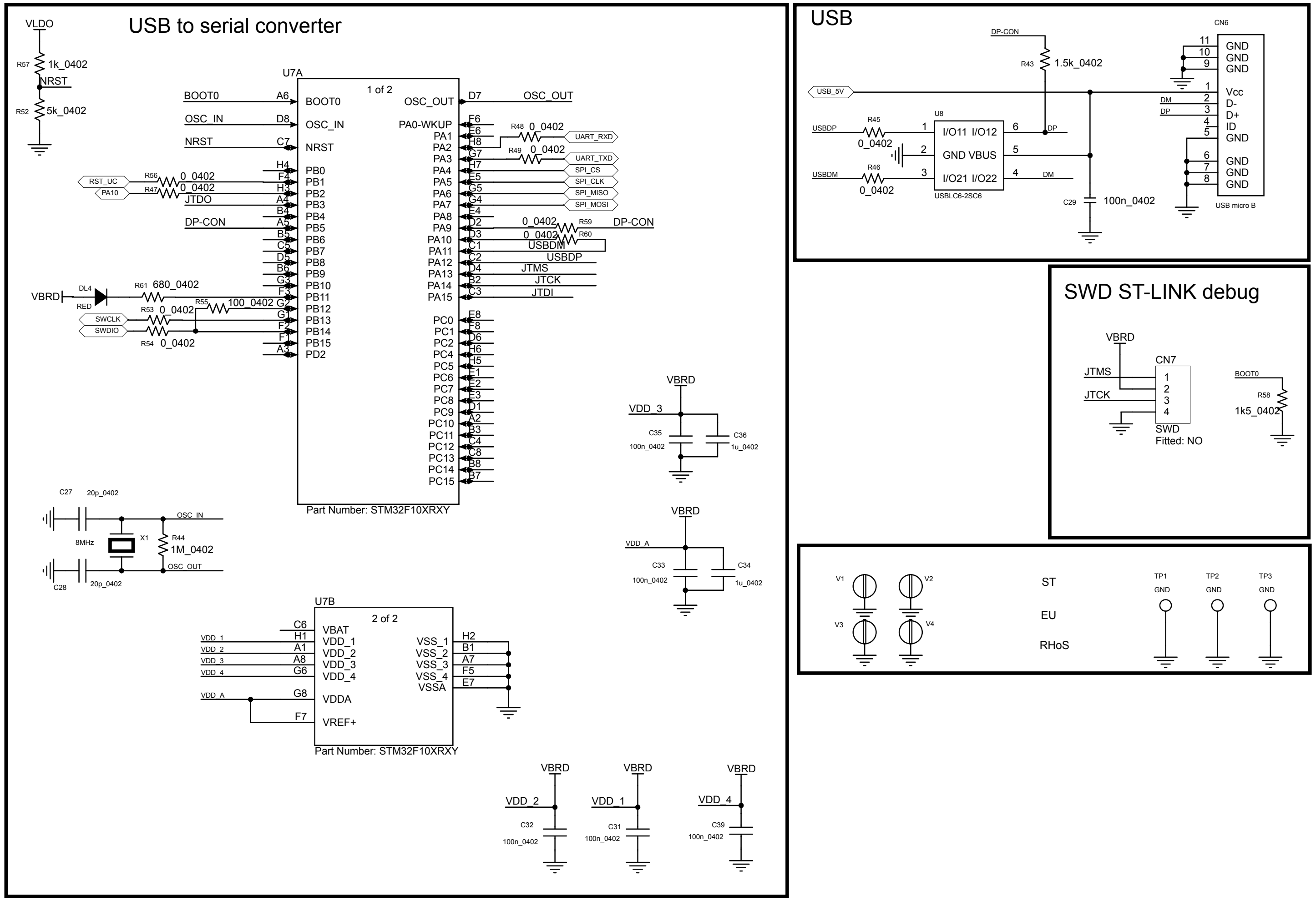


Figure 40. STEVAL-IDB011V2 circuit schematic (3 of 3)



29 STEVAL-IDB010V1 schematic diagrams

Figure 41. STEVAL-IDB010V1 circuit schematic (1 of 7)

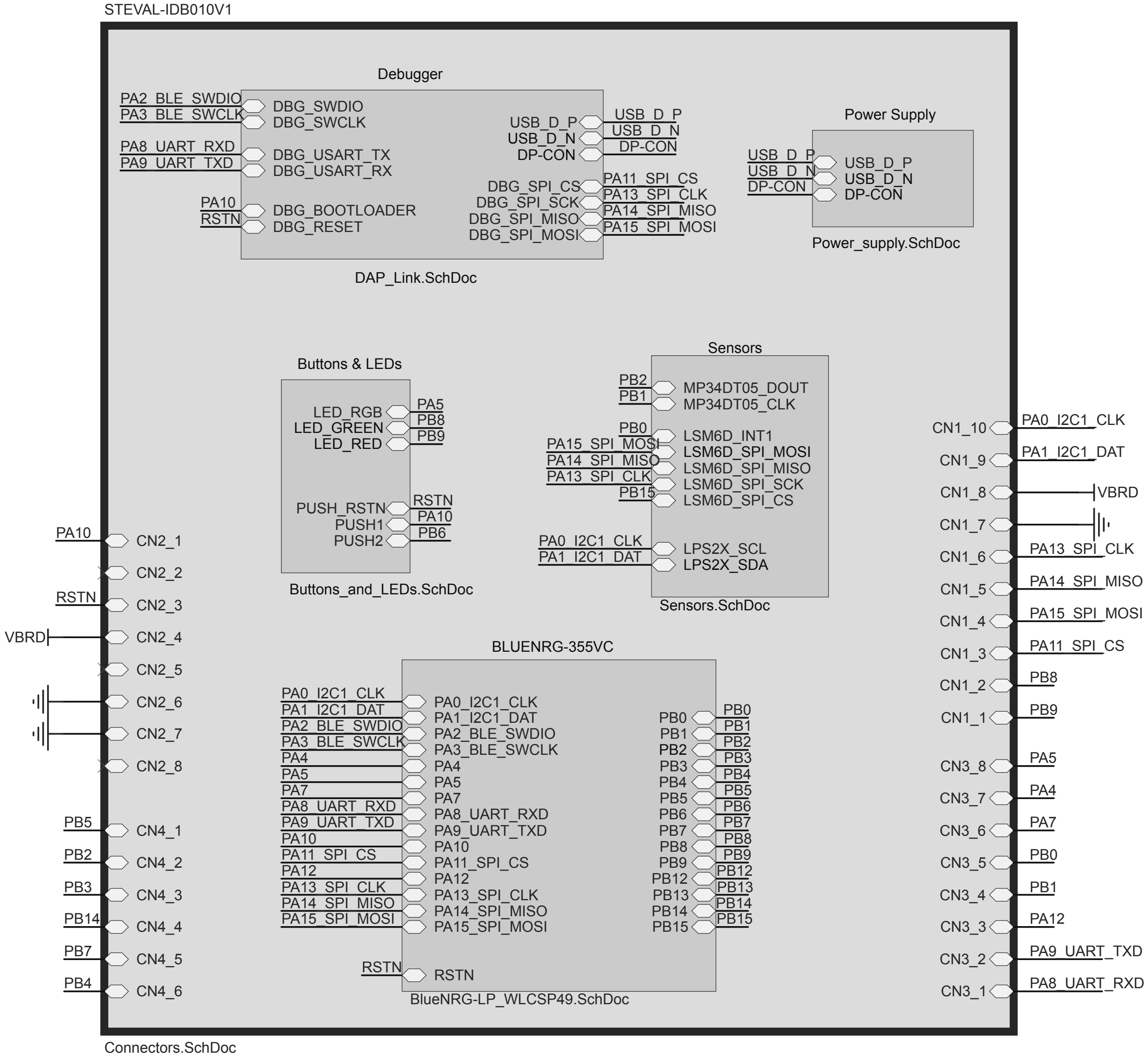


Figure 42. STEVAL-IDB010V1 circuit schematic (2 of 7)

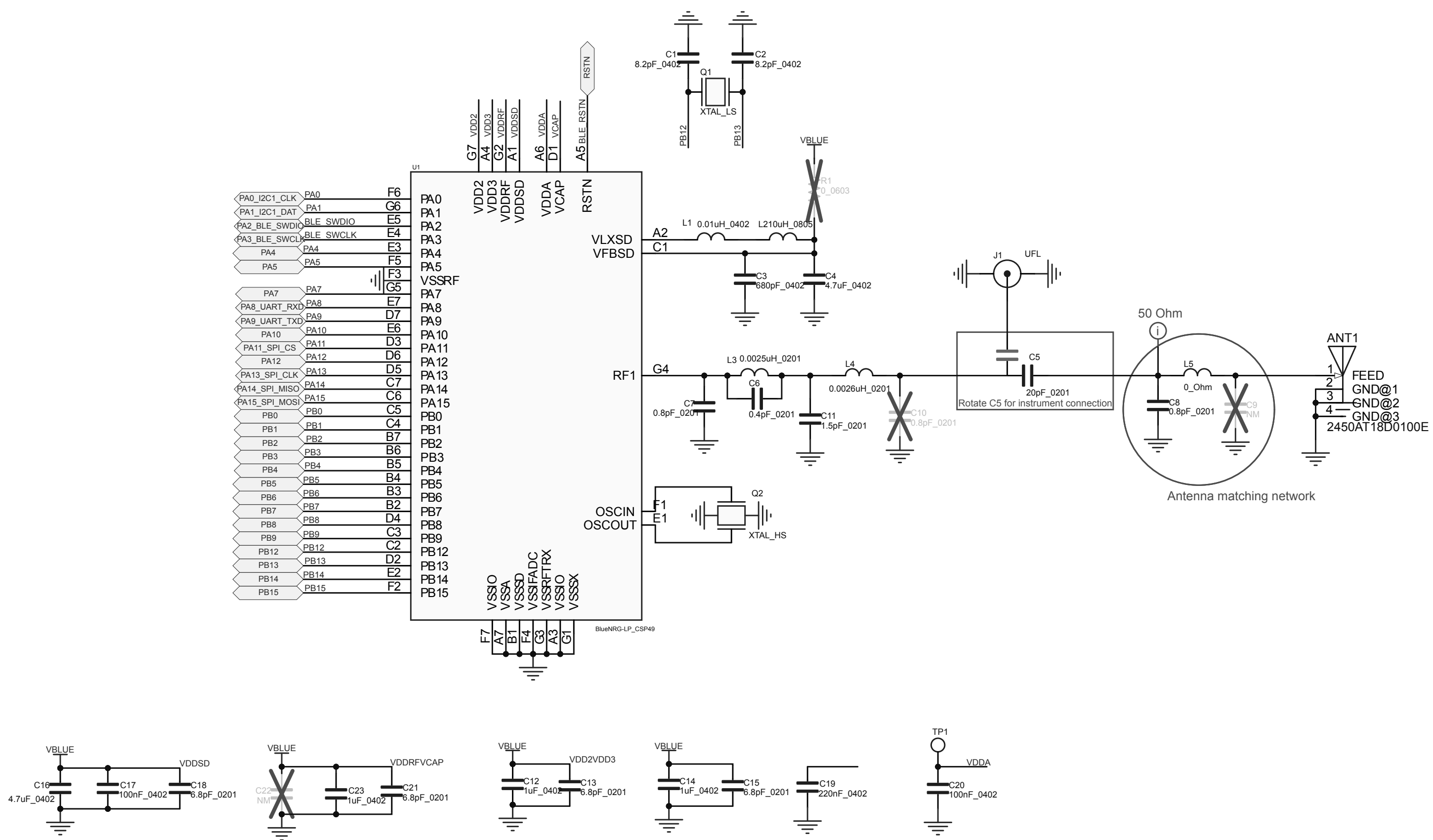


Figure 43. STEVAL-IDB010V1 circuit schematic (3 of 7)

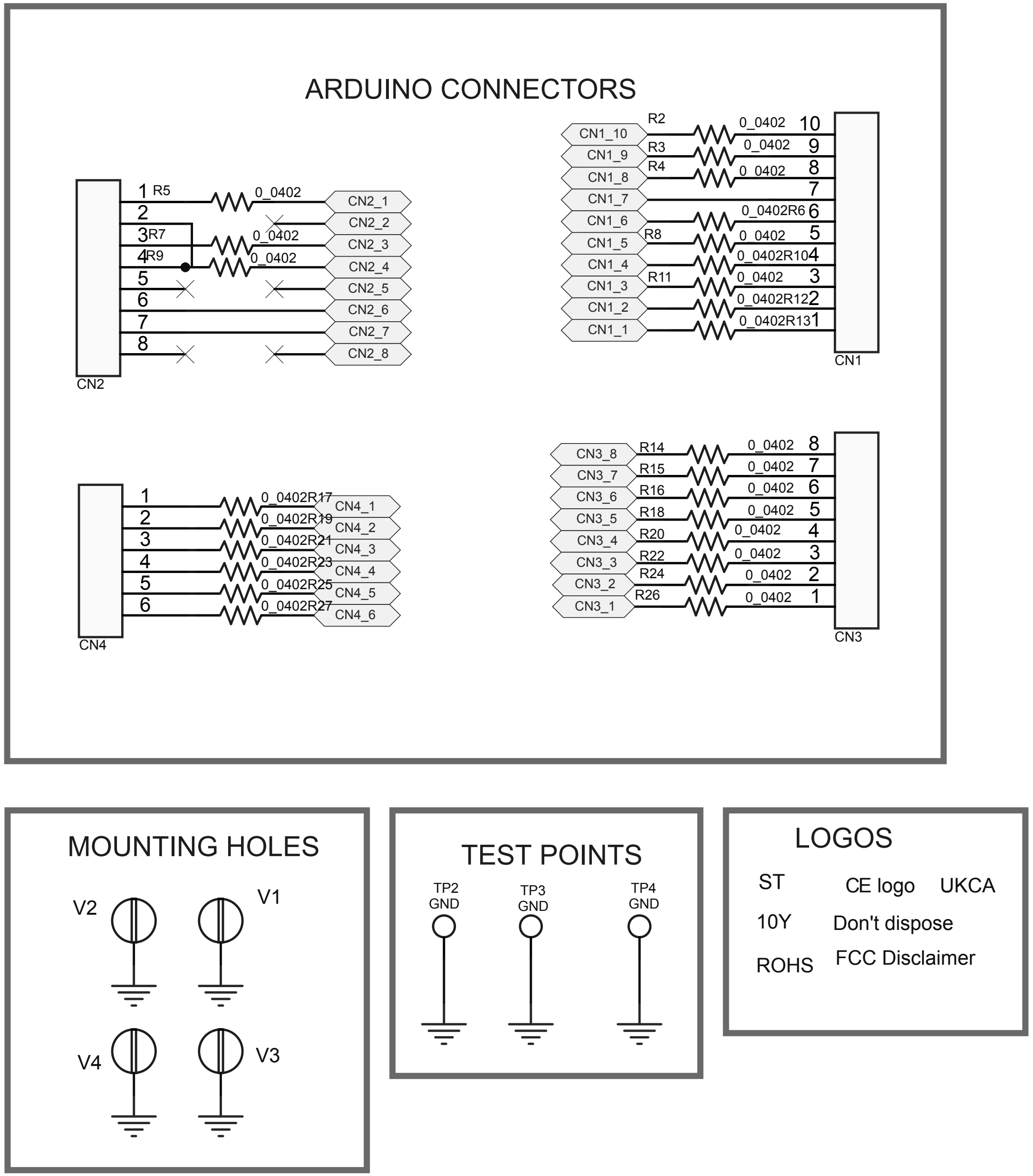


Figure 44. **STEVAL-IDB010V1 circuit schematic (4 of 7)**

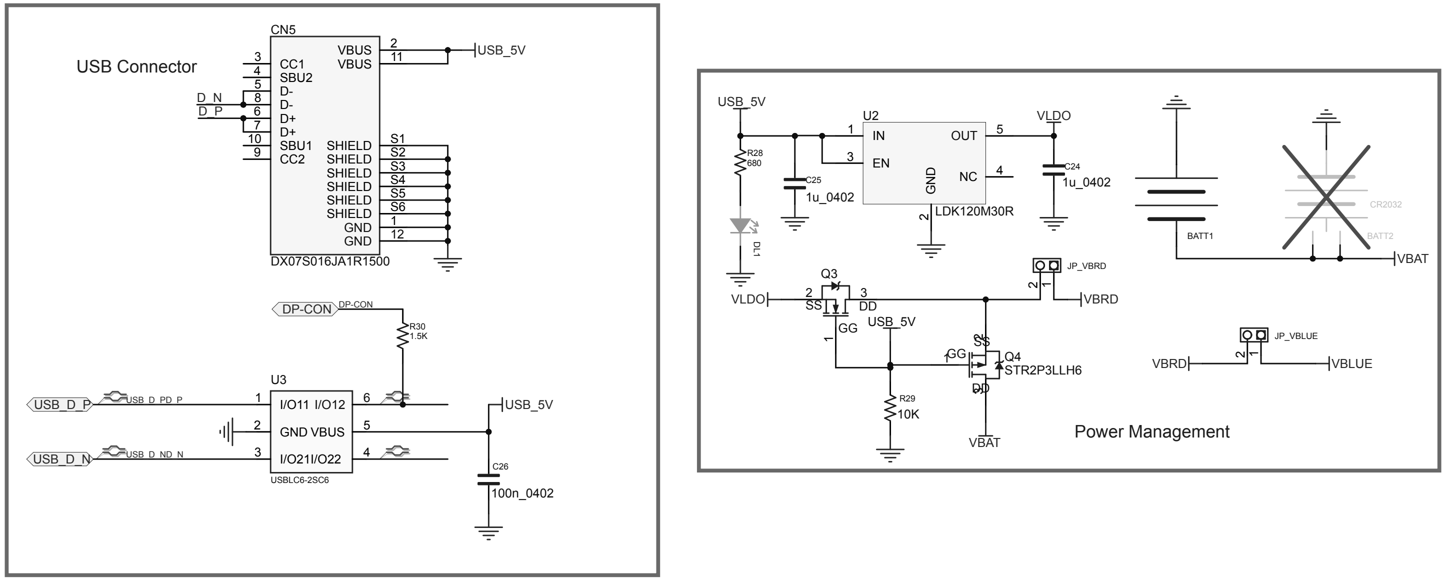


Figure 45. STEVAL-IDB010V1 circuit schematic (5 of 7)

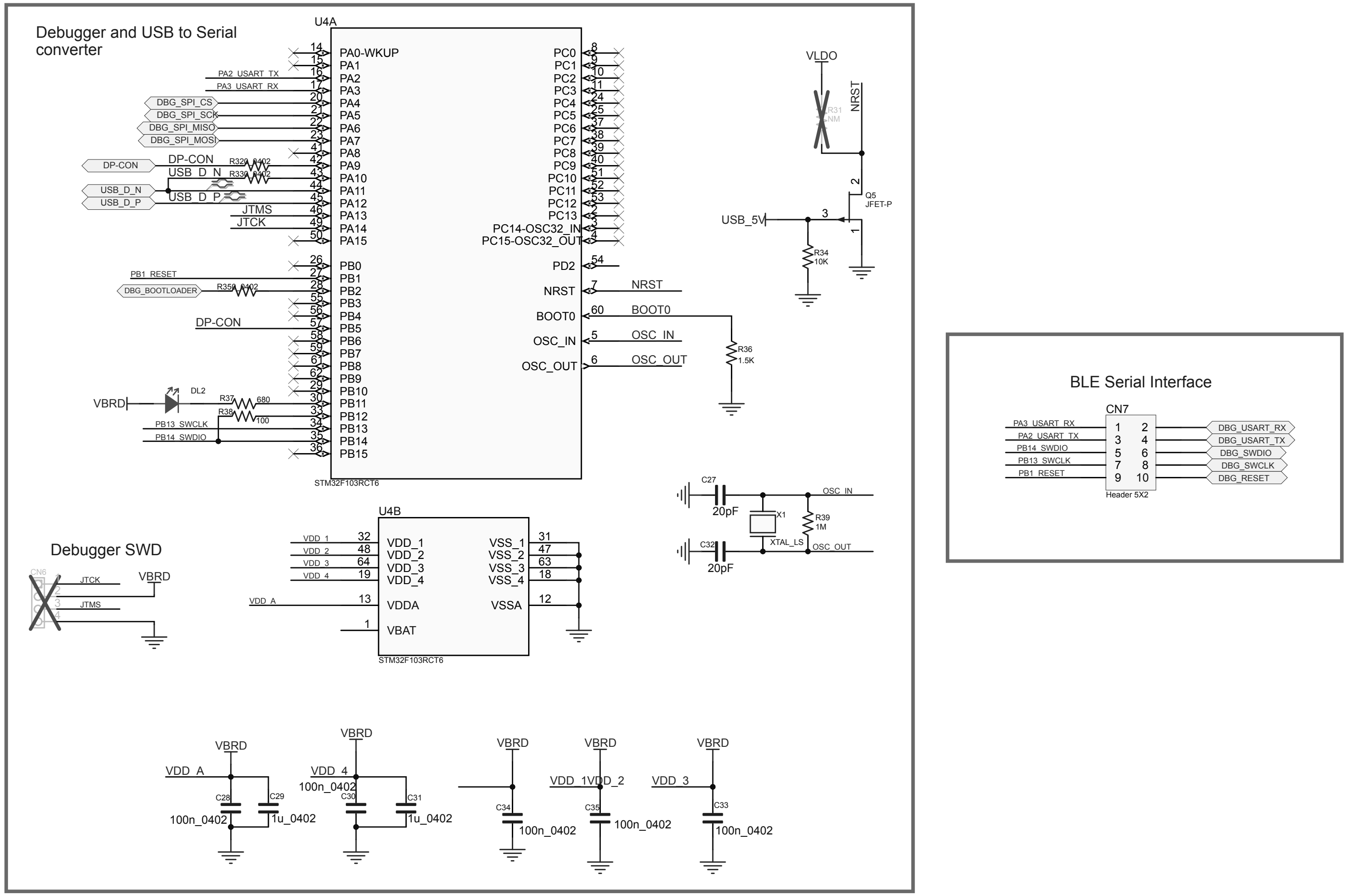


Figure 46. **STEVAL-IDB010V1 circuit schematic (6 of 7)**

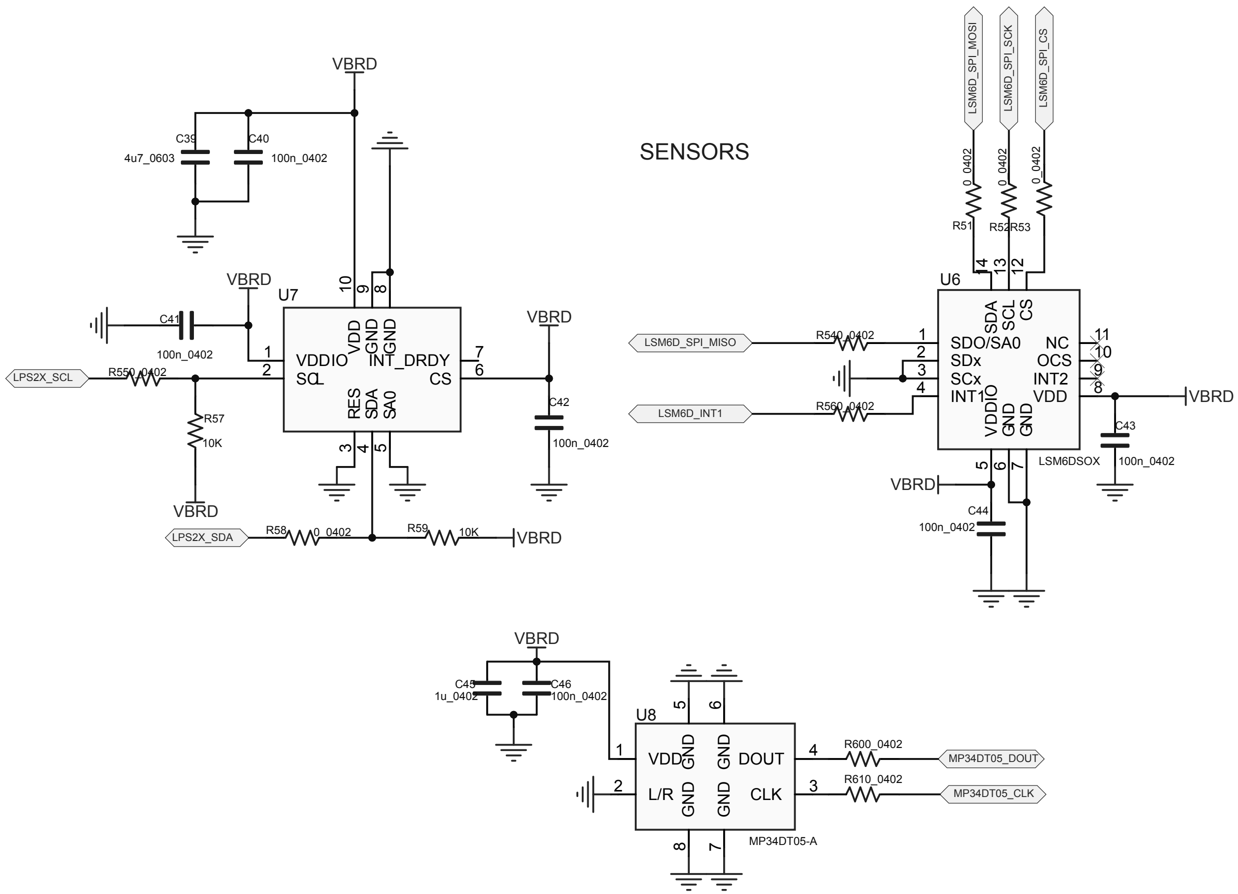
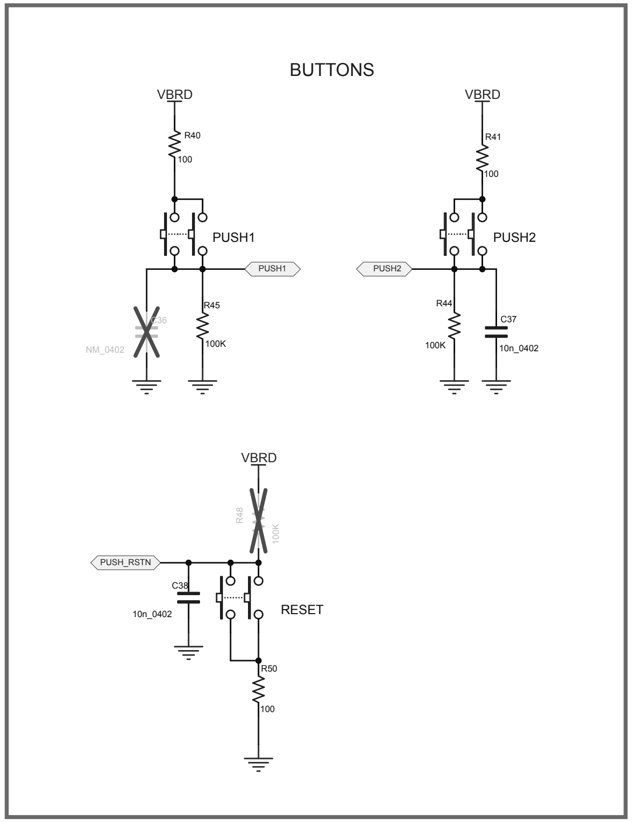
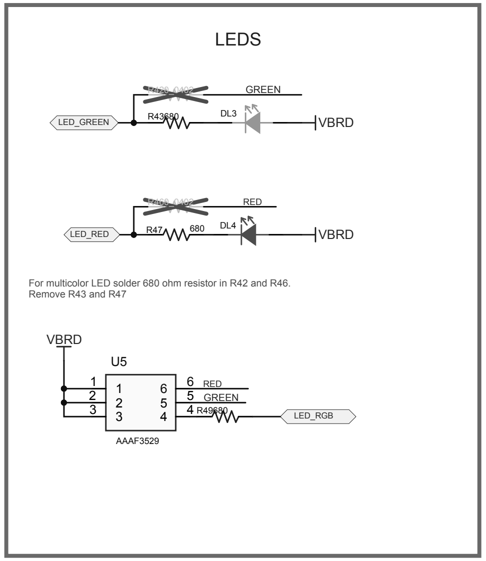


Figure 47. STEVAL-IDB010V1 circuit schematic (7 of 7)





30 STEVAL-IDB012V1 schematic diagrams

Figure 48. STEVAL-IDB012V1 circuit schematic (1 of 7)

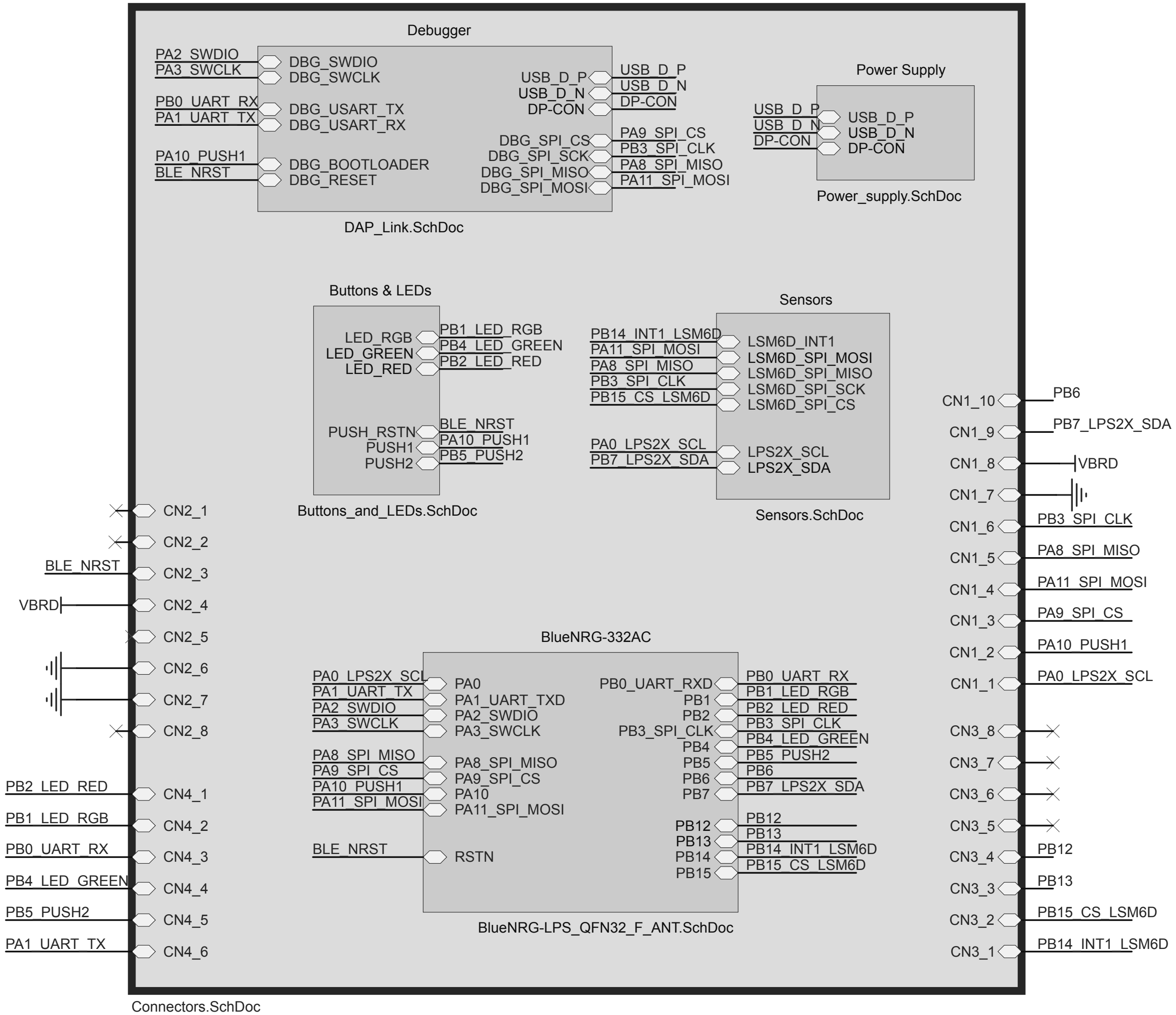


Figure 49. STEVAL-IDB012V1 circuit schematic (2 of 7)

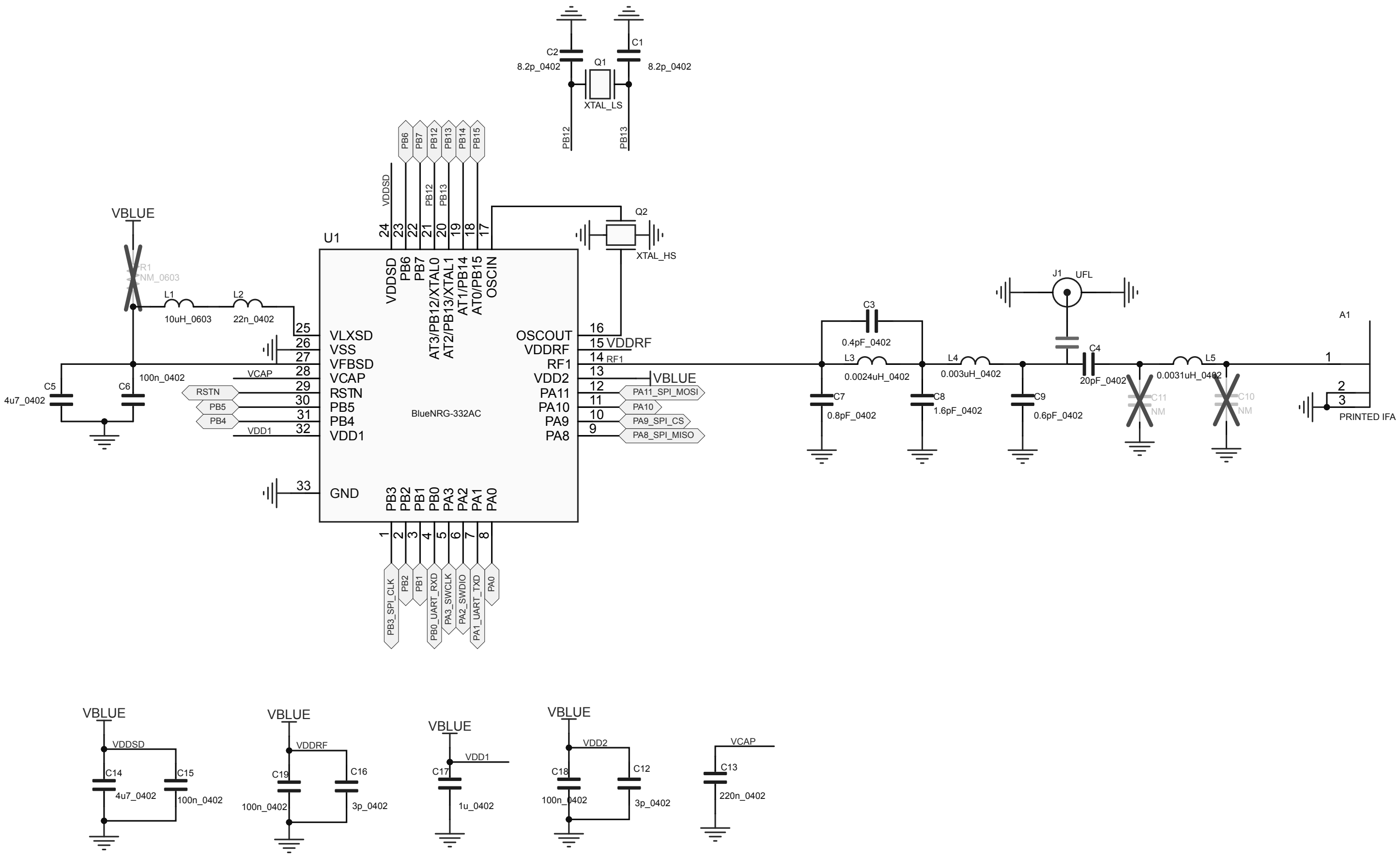


Figure 50. **STEVAL-IDB012V1 circuit schematic (3 of 7)**

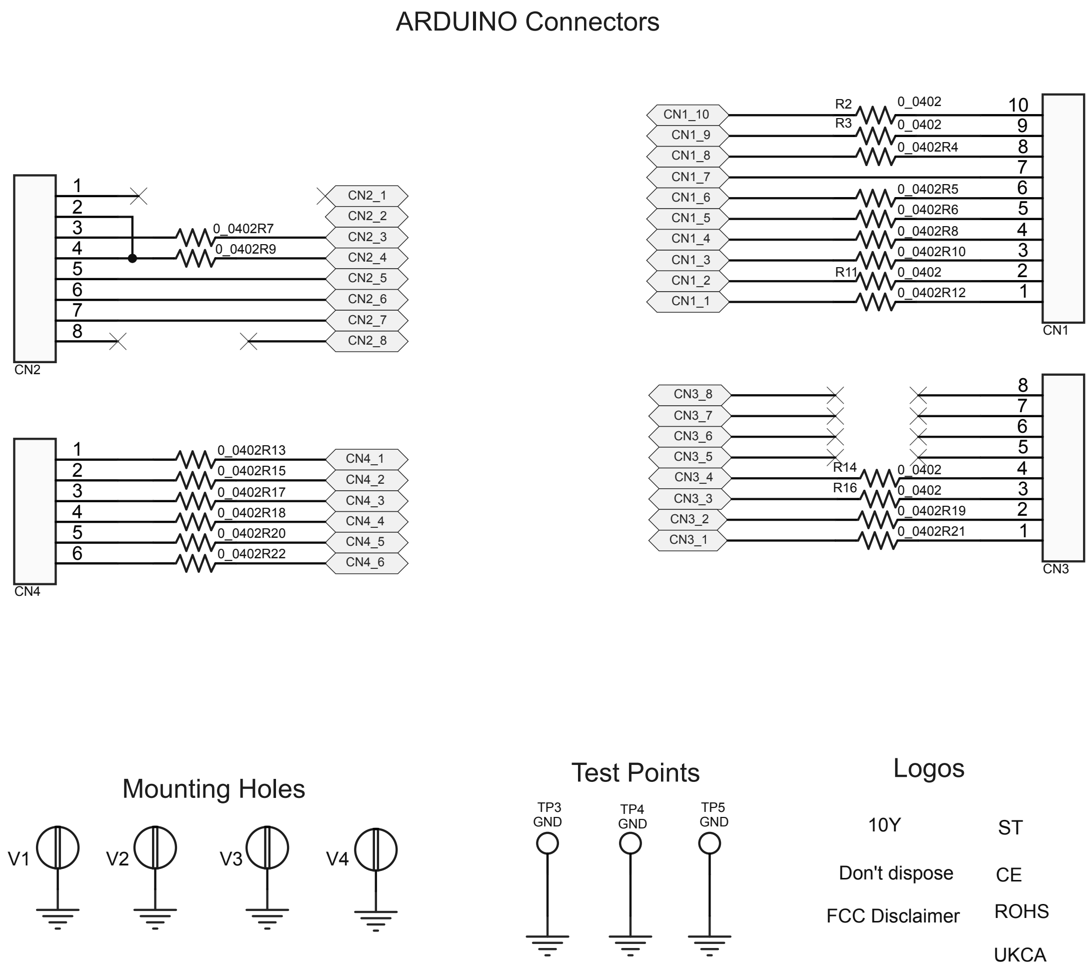


Figure 51. STEVAL-IDB012V1 circuit schematic (4 of 7)

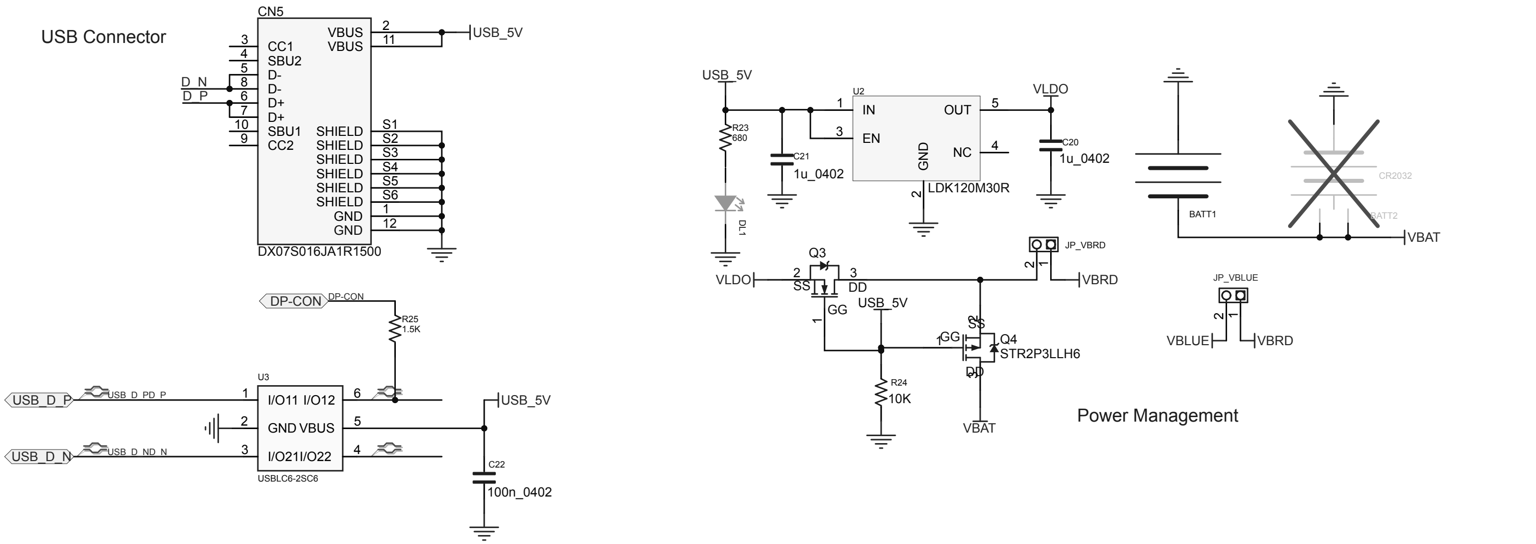


Figure 52. STEVAL-IDB012V1 circuit schematic (5 of 7)

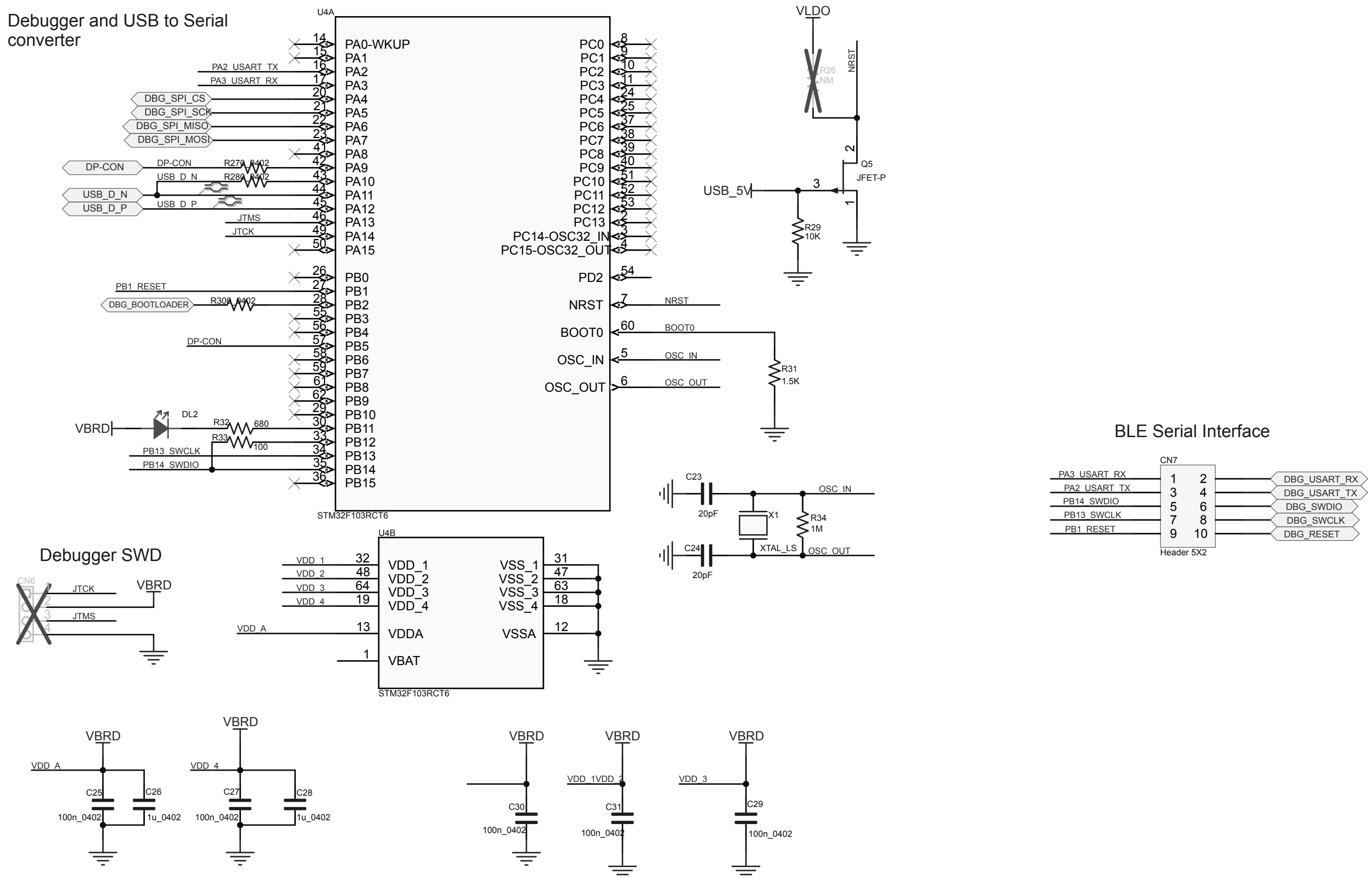


Figure 53. STEVAL-IDB012V1 circuit schematic (6 of 7)

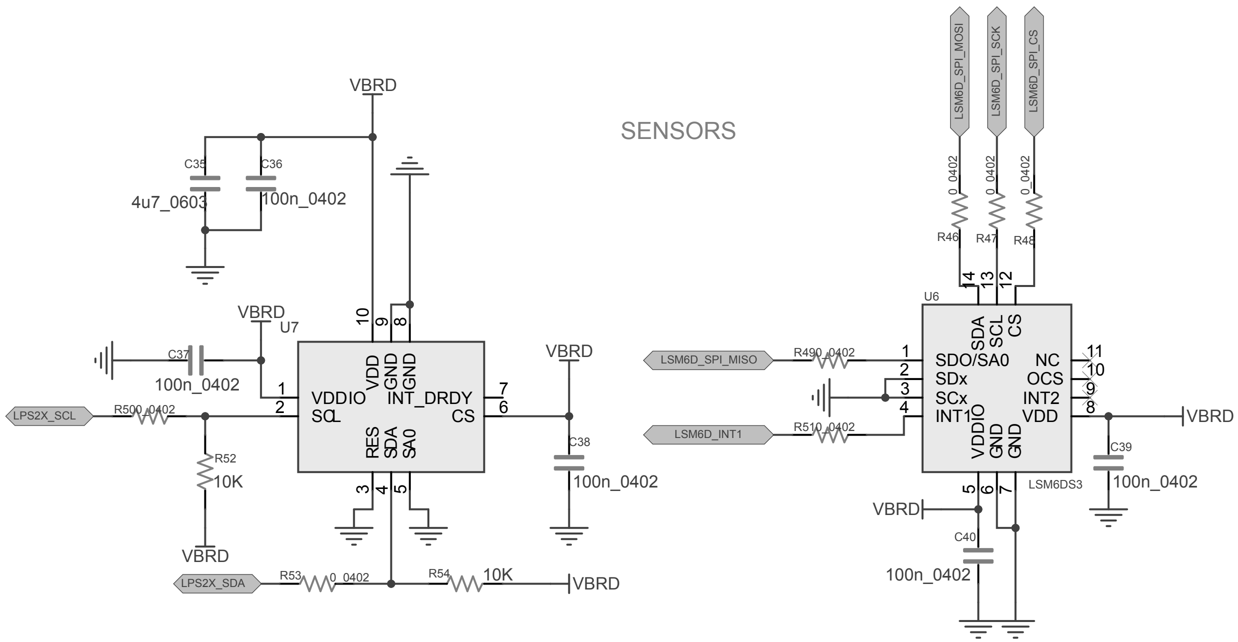
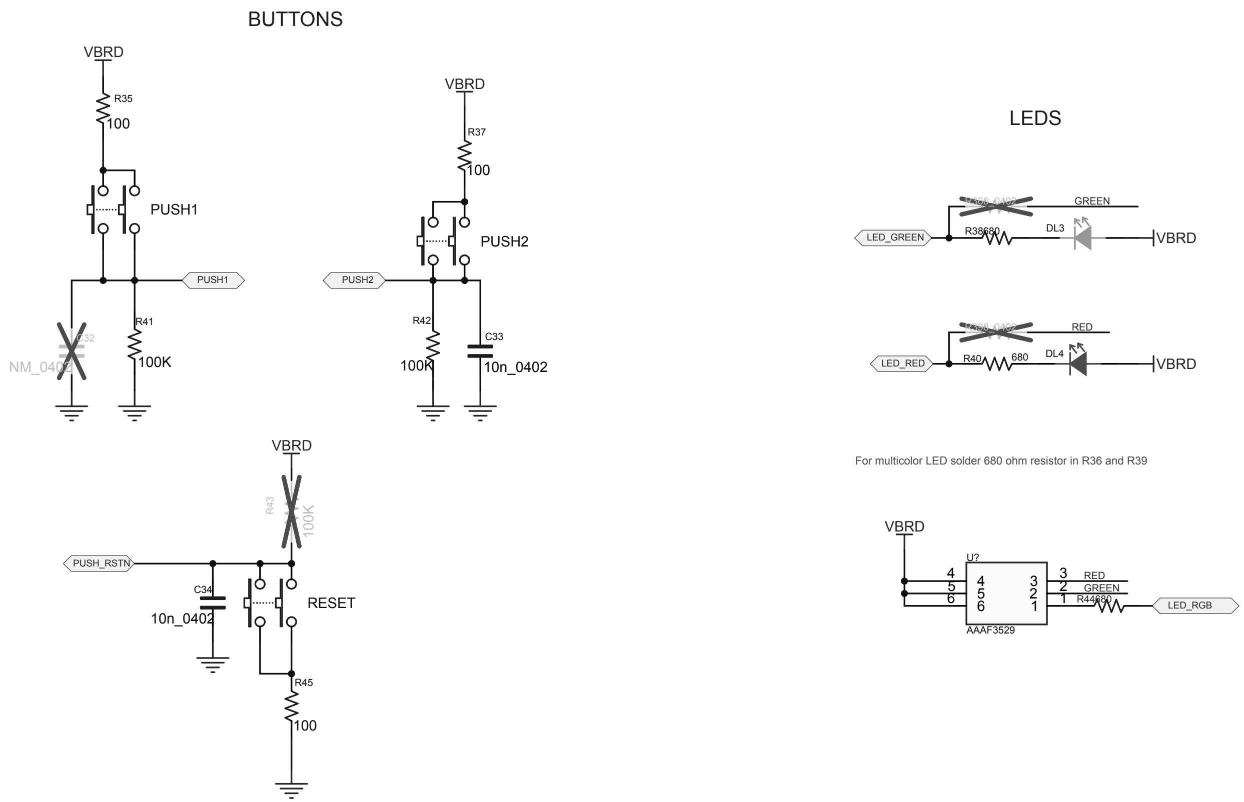


Figure 54. STEVAL-IDB012V1 circuit schematic (7 of 7)



31 STEVAL-IDB013V1 schematic diagrams

Figure 55. STEVAL-IDB013V1 circuit schematic (1 of 7)

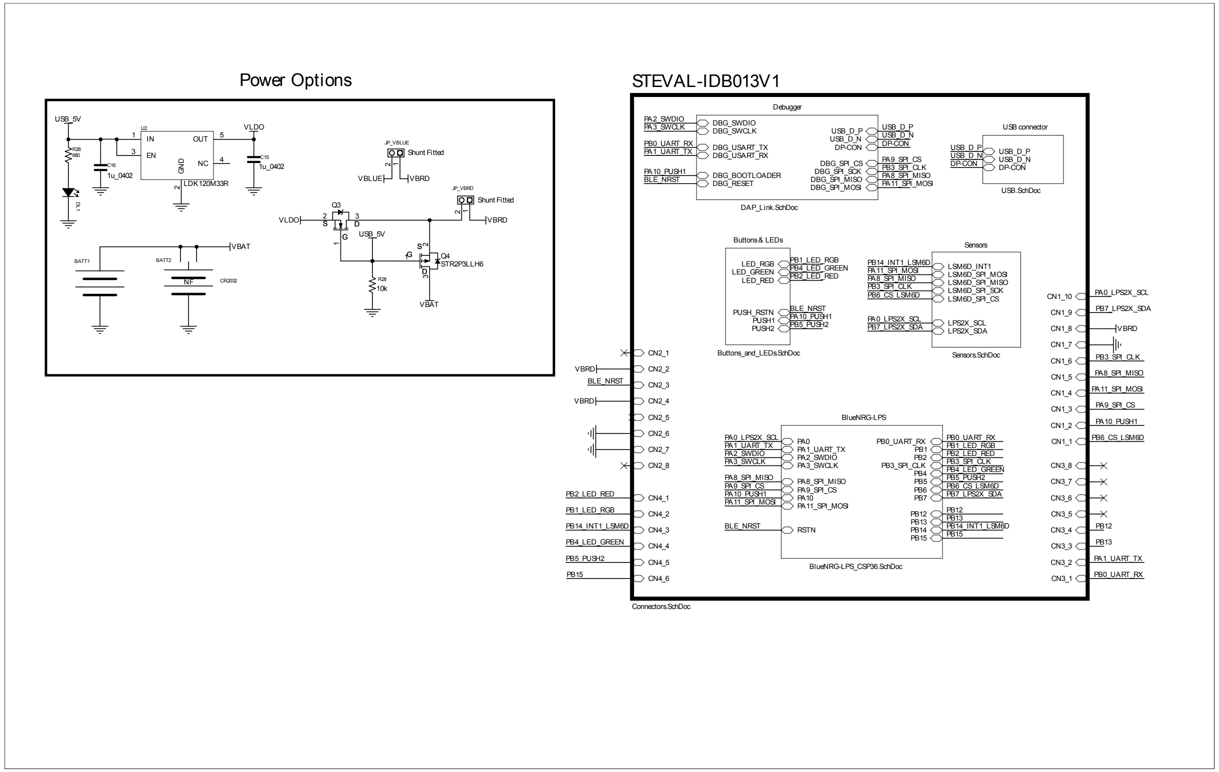


Figure 56. STEVAL-IDB013V1 circuit schematic (2 of 7)

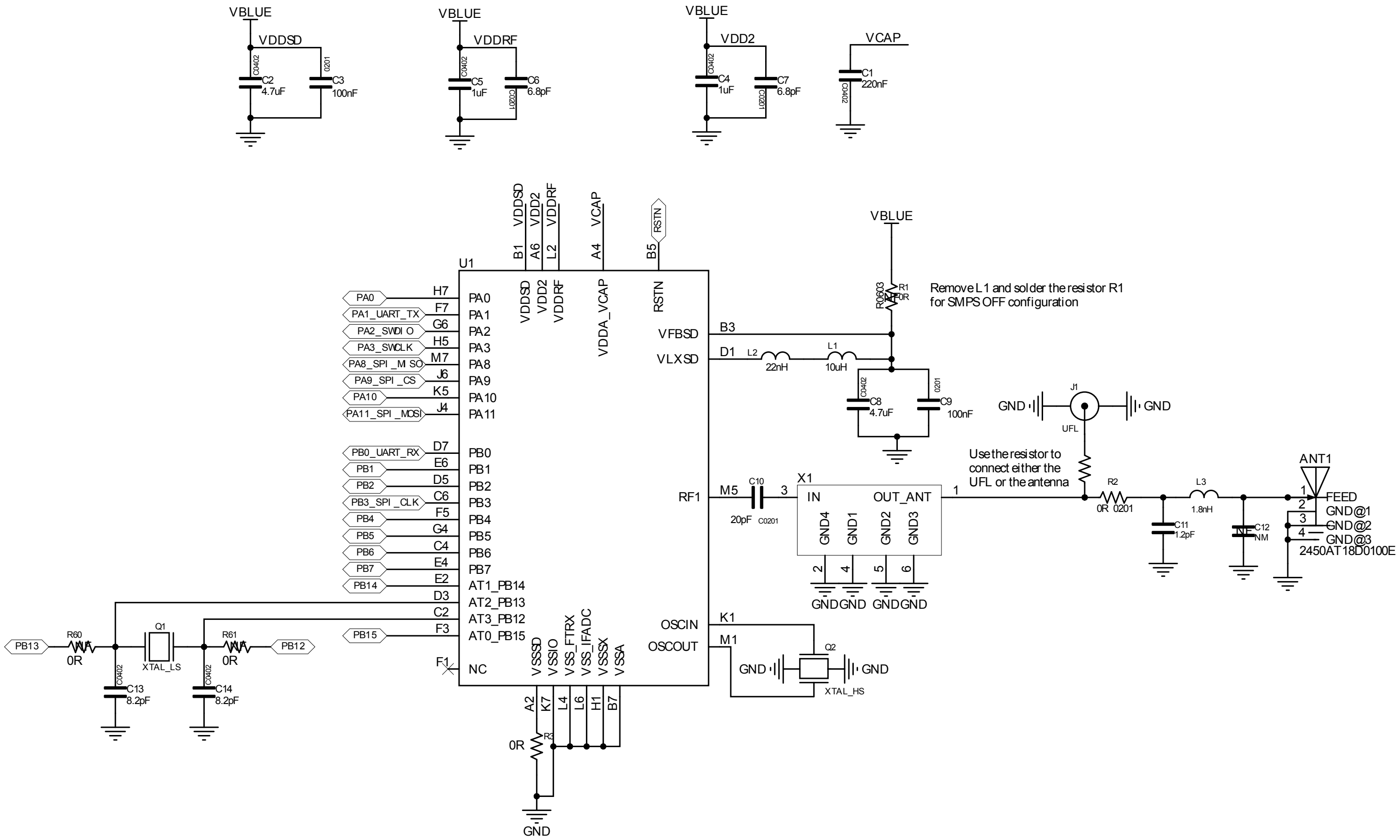


Figure 57. STEVAL-IDB013V1 circuit schematic (3 of 7)

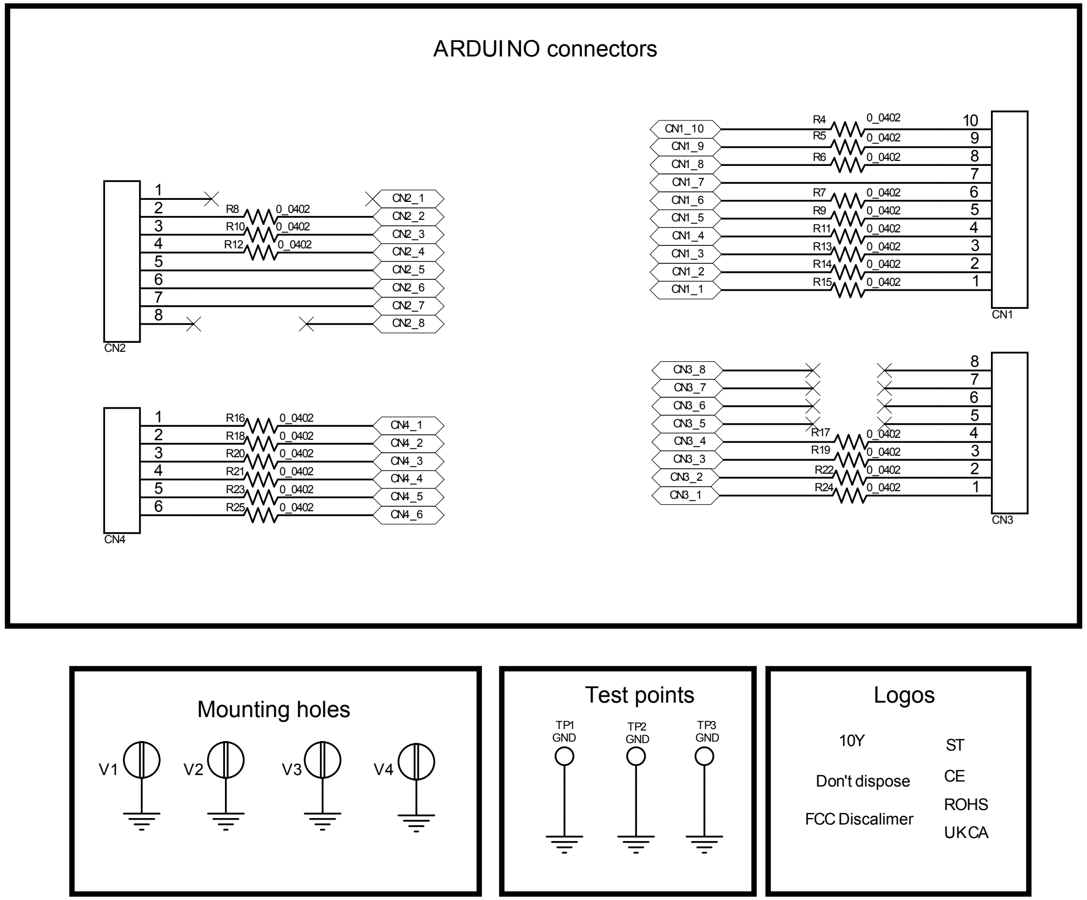


Figure 58. STEVAL-IDB013V1 circuit schematic (4 of 7)

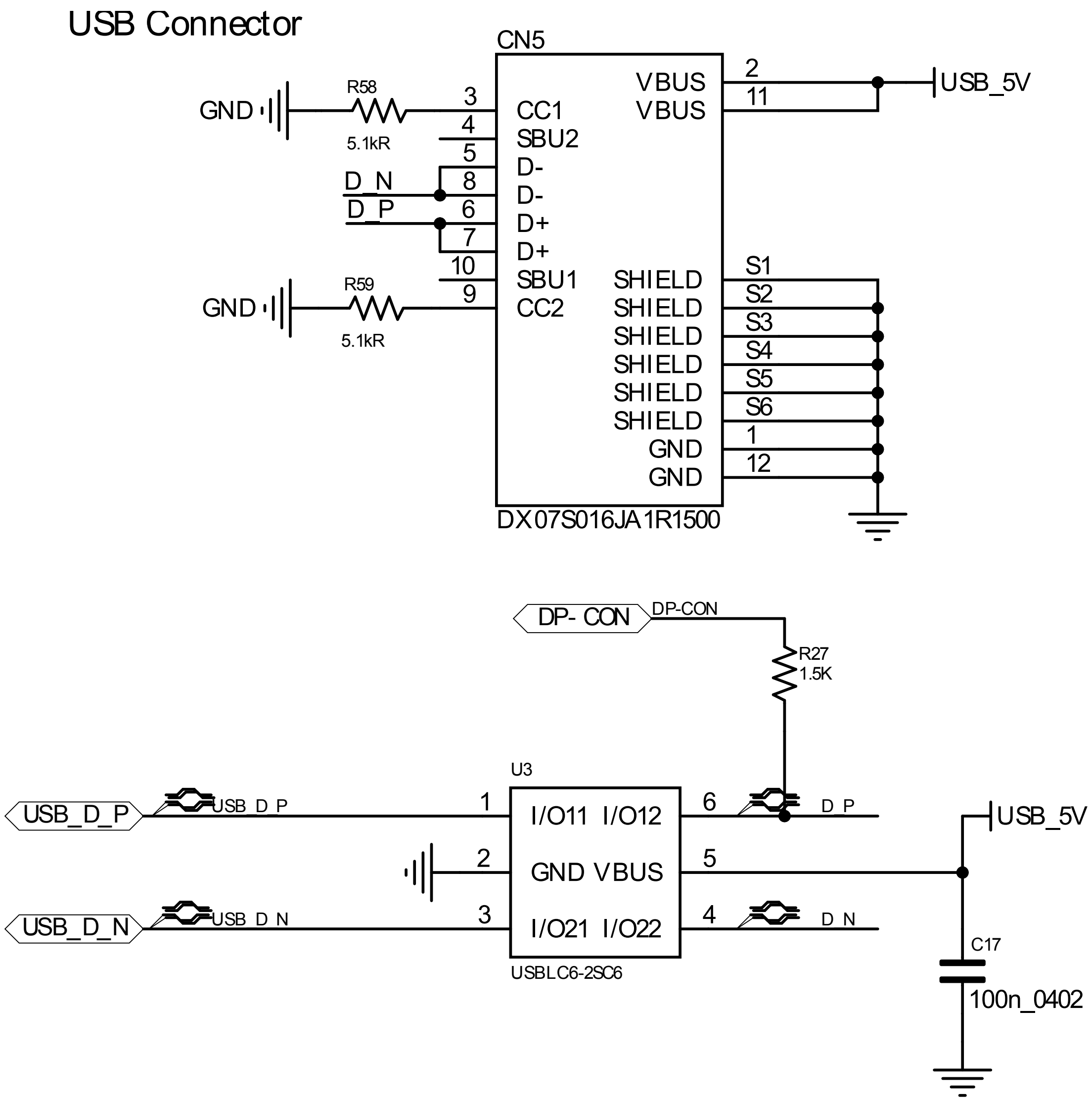


Figure 59. STEVAL-IDB013V1 circuit schematic (5 of 7)

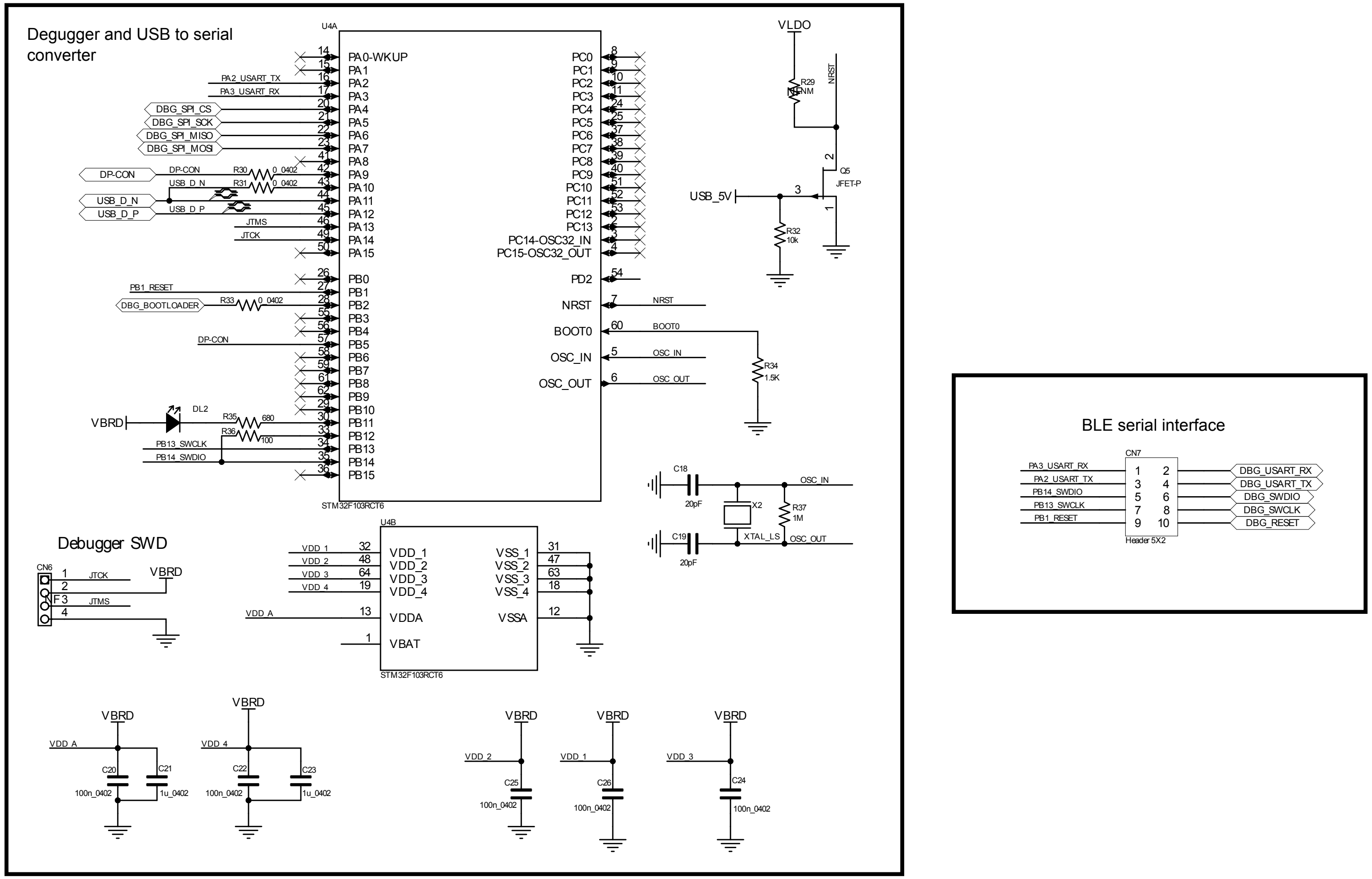


Figure 60. STEVAL-IDB013V1 circuit schematic (6 of 7)

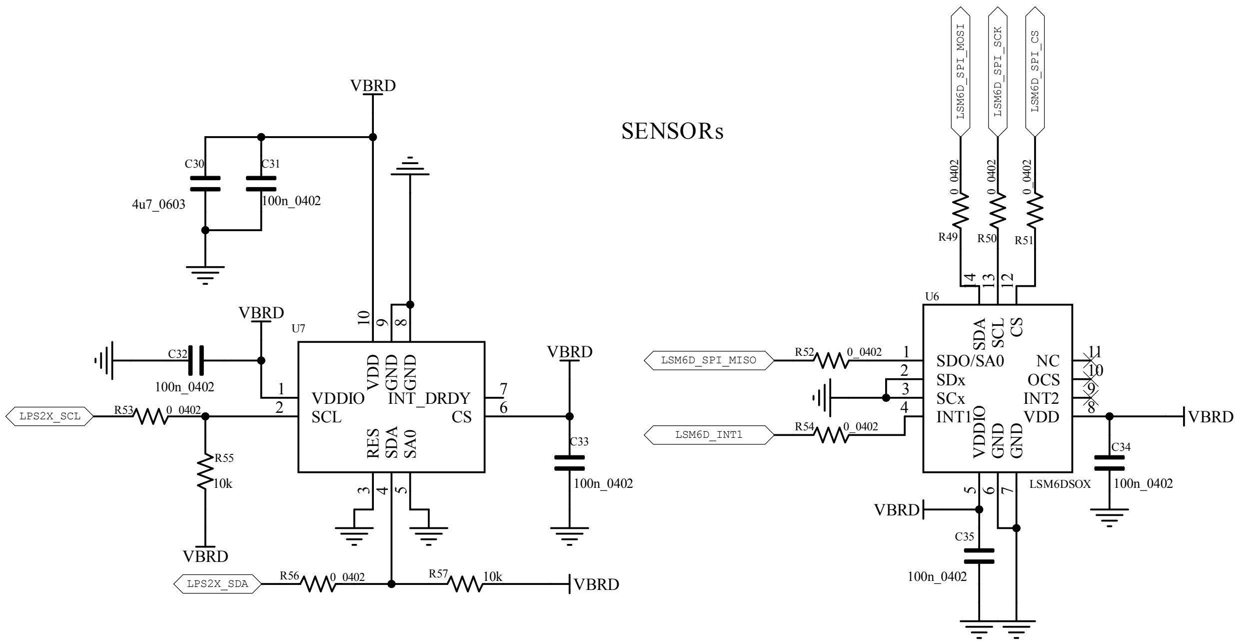
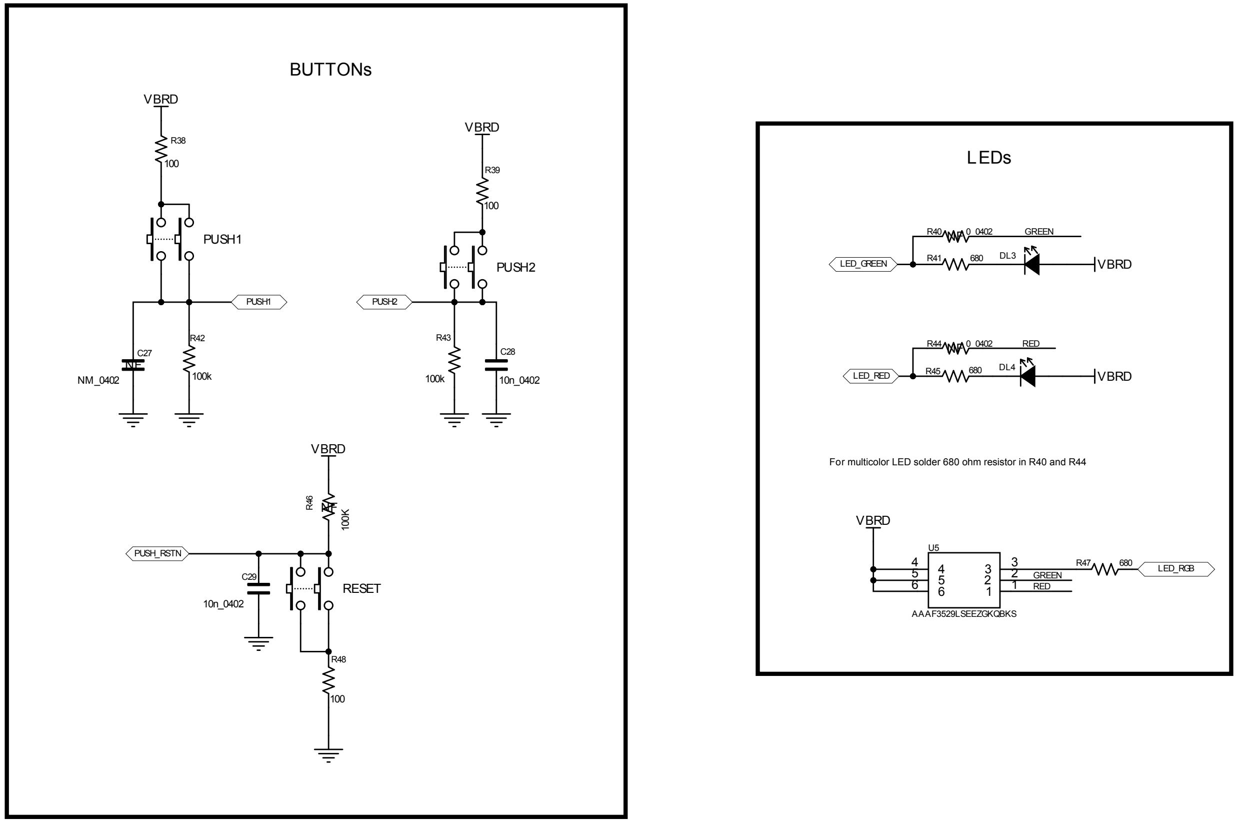


Figure 61. STEVAL-IDB013V1 circuit schematic (7 of 7)



32 STEVAL-IDB011V1 versions

Table 21. STEVAL-IDB011V1 versions

Finished good	Schematic diagrams	Bill of materials
STEVAL$IDB011V1A (1)	STEVAL$IDB011V1A schematic diagrams	STEVAL$IDB011V1A bill of materials

1. *This code identifies the STEVAL-IDB011V1 evaluation kit first version.*

33 STEVAL-IDB011V2 versions

Table 22. **STEVAL-IDB011V2 versions**

Finished good	Schematic diagrams	Bill of materials
STEVAL$IDB011V2A [1]	STEVAL$IDB011V2A schematic diagrams	STEVAL$IDB011V2A bill of materials

1. *This code identifies the STEVAL-IDB011V2 evaluation kit first version.*

34 STEVAL-IDB010V1 versions

Table 23. STEVAL-IDB010V1 versions

PCB version	Schematic diagrams	Bill of materials
STEVAL$IDB010V1A	STEVAL$IDB010V1A schematic diagrams	STEVAL$IDB010V1A bill of materials

1. *This code identifies the STEVAL-IDB010V1 evaluation kit first version. It is printed on the board PCB.*

35 STEVAL-IDB012V1 versions

Table 24. STEVAL-IDB012V1 versions

PCB version	Schematic diagrams	Bill of materials
STEVAL$IDB012V1A	STEVAL$IDB012V1A schematic diagrams	STEVAL$IDB012V1A bill of materials

1. *This code identifies the STEVAL-IDB012V1 evaluation kit first version. It is printed on the board PCB.*

36 STEVAL-IDB013V1 versions

Table 25. **STEVAL-IDB013V1 versions**

PCB version	Schematic diagrams	Bill of materials
STEVAL$IDB013V1A	STEVAL$IDB013V1A schematic diagrams	STEVAL$IDB013V1A bill of materials

1. *This code identifies the STEVAL-IDB013V1 evaluation kit first version. It is printed on the board PCB.*

37 STEVAL-IDB011V2 regulatory compliance

Formal Notice Required by the U.S. Federal Communications Commission

FCC NOTICE:

This kit is designed to allow:

(1) Product developers to evaluate electronic components, circuitry, or software associated with the kit to determine

whether to incorporate such items in a finished product and

(2) Software developers to write software applications for use with the end product.

This kit is not a finished product and when assembled may not be resold or otherwise marketed unless all required FCC equipment authorizations are first obtained. Operation is subject to the condition that this product not cause harmful interference to licensed radio stations and that this product accept harmful interference. Unless the assembled kit is designed to operate under part 15, part 18 or part 95 of this chapter, the operator of the kit must operate under the authority of an FCC license holder or must secure an experimental authorization under part 5 of this chapter 3.1.2.

Formal Product Notice Required by Industry Canada Innovation, Science and Economic Development

Canada compliance:

For evaluation purposes only. This kit generates, uses, and can radiate radio frequency energy and has not been tested for compliance with the limits of computing devices pursuant to Industry Canada (IC) rules.

À des fins d'évaluation uniquement. Ce kit génère, utilise et peut émettre de l'énergie radiofréquence et n'a pas été testé pour sa conformité aux limites des appareils informatiques conformément aux règles d'Industrie Canada (IC).

Formal product notice required by EU

The STEVAL-IDB011V2 kit is in conformity with the essential requirements of the Directive 2014/53/EU (RED) and of the Directive 2015/863/EU (RoHS). Harmonized standards applied are listed in the EU Declaration of Conformity.

38 STEVAL-IDB010V1 regulatory compliance

Formal Notice Required by the U.S. Federal Communications Commission

FCC NOTICE:

This kit is designed to allow:

(1) Product developers to evaluate electronic components, circuitry, or software associated with the kit to determine whether to incorporate such items in a finished product and

(2) Software developers to write software applications for use with the end product.

This kit is not a finished product and when assembled may not be resold or otherwise marketed unless all required FCC equipment authorizations are first obtained. Operation is subject to the condition that this product not cause harmful interference to licensed radio stations and that this product accept harmful interference. Unless the assembled kit is designed to operate under part 15, part 18 or part 95 of this chapter, the operator of the kit must operate under the authority of an FCC license holder or must secure an experimental authorization under part 5 of this chapter 3.1.2.

Formal Product Notice Required by Industry Canada Innovation, Science and Economic Development

Canada compliance:

For evaluation purposes only. This kit generates, uses, and can radiate radio frequency energy and has not been tested for compliance with the limits of computing devices pursuant to Industry Canada (IC) rules.

À des fins d'évaluation uniquement. Ce kit génère, utilise et peut émettre de l'énergie radiofréquence et n'a pas été testé pour sa conformité aux limites des appareils informatiques conformément aux règles d'Industrie Canada (IC).

Formal product notice required by EU

The kit STEVAL-IDB010V1 is in conformity with the essential requirements of the Directive 2014/53/EU (RED) and of the Directive 2015/863/EU (RoHS). Harmonized standards applied are listed in the EU Declaration of Conformity.

Formal product notice required by Great Britain

The kit STEVAL-IDB010V1 is in compliance with the UK Radio Equipment Regulations 2017. The full text of the UK declaration of conformity is available at the following internet address: www.st.com/en/evaluation-tools/steval-idb010v1.html

39 STEVAL-IDB012V1 regulatory compliance

Formal Notice Required by the U.S. Federal Communications Commission

FCC NOTICE:

This kit is designed to allow:

(1) Product developers to evaluate electronic components, circuitry, or software associated with the kit to determine

whether to incorporate such items in a finished product and

(2) Software developers to write software applications for use with the end product.

This kit is not a finished product and when assembled may not be resold or otherwise marketed unless all required FCC equipment authorizations are first obtained. Operation is subject to the condition that this product not cause harmful interference to licensed radio stations and that this product accept harmful interference. Unless the assembled kit is designed to operate under part 15, part 18 or part 95 of this chapter, the operator of the kit must operate under the authority of an FCC license holder or must secure an experimental authorization under part 5 of this chapter 3.1.2.

Formal Product Notice Required by Industry Canada Innovation, Science and Economic Development

Canada compliance:

For evaluation purposes only. This kit generates, uses, and can radiate radio frequency energy and has not been tested for compliance with the limits of computing devices pursuant to Industry Canada (IC) rules.

À des fins d'évaluation uniquement. Ce kit génère, utilise et peut émettre de l'énergie radiofréquence et n'a pas été testé pour sa conformité aux limites des appareils informatiques conformément aux règles d'Industrie Canada (IC).

Formal product notice required by EU

The kit STEVAL-IDB012V1 is in conformity with the essential requirements of the Directive 2014/53/EU (RED) and of the Directive 2015/863/EU (RoHS). Harmonized standards applied are listed in the EU Declaration of Conformity.

Formal product notice required by Great Britain

The kit STEVAL-IDB012V1 is in compliance with the UK Radio Equipment Regulations 2017. The full text of the UK declaration of conformity is available at the following internet address: www.st.com/en/evaluation-tools/steval-idb012v1.html.

40 STEVAL-IDB013V1 regulatory compliance

Formal Notice Required by the U.S. Federal Communications Commission

For evaluation only; not FCC approved for resale

FCC NOTICE:

This kit is designed to allow:

(1)

Product developers to evaluate electronic components, circuitry, or software associated with the kit to determine whether to incorporate such items in a finished product and

(2) Software developers to write software applications for use with the end product.

This kit is not a finished product and when assembled may not be resold or otherwise marketed unless all required FCC equipment authorizations are first obtained. Operation is subject to the condition that this product not cause harmful interference to licensed radio stations and that this product accept harmful interference. Unless the assembled kit is designed to operate under part 15, part 18 or part 95 of this chapter, the operator of the kit must operate under the authority of an FCC license holder or must secure an experimental authorization under part 5 of this chapter 3.1.2.

Formal Product Notice Required by Industry Canada Innovation, Science and Economic Development

Canada compliance:

For evaluation purposes only. This kit generates, uses, and can radiate radio frequency energy and has not been tested for compliance with the limits of computing devices pursuant to Industry Canada (IC) rules.

À des fins d'évaluation uniquement. Ce kit génère, utilise et peut émettre de l'énergie radiofréquence et n'a pas été testé pour sa conformité aux limites des appareils informatiques conformément aux règles d'Industrie Canada (IC).

Formal product notice required by EU

The kit STEVAL-IDB013V1 is in conformity with the essential requirements of the Directive 2014/53/EU (RED) and of the Directive 2015/863/EU (RoHS). Harmonized standards applied are listed in the EU Declaration of Conformity.

Formal product notice required by Great Britain

The kit STEVAL-IDB013V1 is in compliance with the UK Radio Equipment Regulations 2017. The full text of the UK declaration of conformity is available at the following internet address: www.st.com/en/evaluation-tools/steval-idb012v1.html.

Revision history

Table 26. **Document revision history**

Date	Version	Changes
17-Jul-2020	1	Initial release.
29-Jul-2020	2	Updated Figure 23. STEVAL-IDB011V1 circuit schematic (1 of 3).
03-Nov-2020	3	Removed Section 10.1 BlueNRG app for smartphones. Updated Section 10.1 BLE sensor profile demo: connection with a central device, Section 10.1.4 Connection with central device, Section 12 BLE sensor profile central demo and Section 14.1 BLE throughput setup. Added Section 11 BLE Sensor for ST BLE Sensor app and Section 11.1 How to run the ST BLE Sensor app for smartphones.
30-Nov-2020	4	Updated Section 22 Schematic diagrams.
12-May-2021	5	Updated Section 1.2 System requirements, Section 1.3 BlueNRG-LP development kit setup, Section 2.9.3 USB_CMSISDAP programming/debugging feature, Section 2.10 BlueNRG-LP programming and debugging, Section 3 BlueNRG-LP Navigator, Section 3.4 Peripheral driver examples, Section 4.1 How to run the Radio Init Wizard, Section 6.1 Software directories, Section 24.1 ADC examples, Section 24.3 CRC examples, Section 24.4 DMA examples, Section 24.5 EXTI example, Section 24.6 Flash example, Section 24.9 I2C examples, Section 24.15 Public Key Accelerator (PKA) demonstration application, Section 24.17 2.4 GHz radio proprietary MIX examples, Section 24.18 RCC examples, Section 24.21 SPI examples, Section 24.22 TIM examples and Section 24.23 UART examples. Added Section 5 Secure Bootloader GUI, Section 5.1 Key generation tab, Section 5.2 Create Signed Image tab, Section 5.3 Store Key in OTP tab, Section 22 BLE sync demo application, Section 22.1 Application roles, Section 22.2 Running the application, Section 23 BLE power control demo application, Section 23.1 Application roles, Section 23.2 Running the application and Section 24.16 PWR examples. Added references to WiSE-Studio IDE.
26-May-2021	6	Updated Section 6.1 Software directories, Section 16 BLE notification consumer demonstration application, Section 17.2 Central device and Section 19.1 Application scenario.
02-Aug-2021	7	Updated Section 2.1 STEVAL-IDB011V1 board overview and Section 6.1 Software directories. Added Section 6.2 How to move a demo application from BlueNRG-355xy to BlueNRG-345xy.
01-Feb-2022	8	Updated Introduction, Section 1.2 Kit contents, Section 2.1 STEVAL-IDB011V1 and STEVALIDB011V2 board overview, Section 2.1 STEVAL-IDB011V1 and STEVAL-IDB011V2 board overview, Section 2.2 BlueNRG-LP SoC connections, Section 2.3 Power supply, Section 2.9 Extension connector, and Section 6.2 How to move a demo application from BlueNRG-355xy to BlueNRG-345xy. Added Section 1.1 Safety recommendations, Section 1.1.1 Target audience, Section 1.1.2 How to handle the boards, Section 2.4 Circuit protection, Section 2.5 Operating temperature, Section 2.6 Main transient voltage, Section 26 STEVAL-IDB011V2 schematic diagrams, Section 27 STEVAL-IDB011V1 versions, Section 28 STEVAL-IDB012V1 versions, and Section 29 STEVAL-IDB011V2 regulatory compliance. Added STEVAL-IDB011V2 compatibility information.

Date	Version	Changes
12-Apr-2022	9	Updated document title, introduction, Section 1.2 Kit contents, Section 1.3 System requirements, Section 1.4 BlueNRG-LP/BlueNRG-LPS development kit setup, Section 2.3 BlueNRG-LP SoC connections, Section 2.5 Power supply, Section 2.9 Jumpers, Section 2.10 Sensors, Section 2.11 Extension connector, Section 2.12 Push buttons, Section 2.13 LEDs, Section 2.14 CMSIS-DAP and Virtual COM, Section 2.14.2 System functionality checks, Section 2.14.3 USB_CMSISDAP programming/debugging feature, Section 2.15 BlueNRGLP/BlueNRG-LPS programming and debugging, Section 2.16 Current measurements, Section 2.17 Hardware setup for STEVAL-IDB011V1/IDB011V2 kits, Section 3 BlueNRG-LP/BlueNRG-LPS Navigator, Section 4 BlueNRG-LP/BlueNRG-LPS Radio Init Wizard, Section 4.1 How to run the Radio Init Wizard, Section 4.2 Main user interface window, Section 5 Secure Bootloader GUI, Section 6 Programming with the BlueNRG-LP/BlueNRG-LPS system-on-chip, Section 6.1 Software directories, Section 6.2 How to move a demo application from BlueNRG-355xy to BlueNRG-345xy, Section 7 BLE beacon demonstration application, Section 7.2 BLE Beacon FreeRTOS example, Section 8 BLE Serial port demo application, Section 9 BLE Serial port master and slave demo application, Section 10 BLE remote control demo application, Section 10.1.4 Connection with a Bluetooth® Low Energy Central device, Section 11 BLE sensor profile demo, Section 11.1.2 Add service and characteristics, Section 12 BLE Sensor for STBLESensor app, Section 12.1 How to run the STBLESensor app for smartphones, Section 13 BLE sensor profile central demo, Section 14 BLE HID/HOGP demonstration applications, Section 15 BLE throughput demonstration application, Section 15.1 BLE throughput setup, Section 16 BLE notification consumer demonstration application, Section 17 BLE security demonstration applications, Section 18 BLE RC Long Range demonstration application, Section 21 BLE power consumption demo application, Section 22.1 Application roles, Section 23.1 Application roles, Section 25 BlueNRG-LP/BlueNRG-LPS peripheral driver examples, Section 25.1 ADC examples, Section 25.9 I2C examples, Section 25.21 SPI examples, and Section 25.23 UART examples. Added Section 2.2 STEVAL-IDB012V1 board overview, Section 2.4 BlueNRG-LPS SoC connections, Section 2.18 Hardware setup for STEVAL-IDB012V1 kit, Section 7.1.5 Periodic advertising mode, Section 7.1.6 AoA tag mode, Section 24 BLE direction finding demo application, Section 24.1 Application roles, Section 24.2 Running the application, Section 28 STEVAL-IDB012V1 schematic diagrams, Section 31 STEVAL-IDB012V1 versions, and Section 33 STEVAL-IDB012V1 regulatory compliance.
17-May-2022	10	Updated introduction, Section 1.2 Kit contents, Section 2.4 BlueNRG-LP SoC connections, Section 2.6 Power supply, Section 2.12 Extension connector, Section 2.13 Push buttons, Section 2.14 LEDs, Section 2.15 CMSIS-DAP and Virtual COM, Section 2.15.2 System functionality checks, Section 2.15.3 USB_CMSISDAP programming/debugging feature, Section 2.16 BlueNRG-LP/BlueNRG-LPS programming and debugging, Section 2.17 Current measurements, Section 2.19 Hardware setup for STEVAL-IDB010V1/STEVAL-IDB012V1 kit, Section 3.6 Development kits, Section 6.1 Software directories, Section 7 BLE beacon demonstration application, Section 8 BLE Serial port demo application, Section 9 BLE Serial port master and slave demo application, Section 10 BLE remote control demo application, Section 11 BLE sensor profile demo, Section 12 BLE Sensor for STBLESensor app, Section 12.1 How to run the STBLESensor app for smartphones, Section 13 BLE sensor profile central demo, Section 15 BLE throughput demonstration application, Section 16 BLE notification consumer demonstration application, Section 17 BLE security demonstration applications, Section 18 BLE RC Long Range demonstration application, Section 21 BLE power consumption demo application, Section 22.1 Application roles, and Section 23.1 Application roles. Added Section 2.2 STEVAL-IDB010V1 board overview, Section 28 STEVAL-IDB010V1 schematic diagrams, Section 32 STEVAL-IDB010V1 versions, and Section 35 STEVAL-IDB010V1 regulatory compliance.
17-Jun-2022	11	Updated introduction, Section 1.2 Kit contents, Section 2.4 BlueNRG-LP SoC connections, and Section 2.6 Power supply.
30-Mar-2023	12	Updated Section 2.2 STEVAL-IDB010V1 board overview, Section 2.3 STEVAL-IDB012V1 and STEVAL-IDB013V1 board overview and Section 3.6 Development kits. Added Section 2.7 Physical layer testing and Section 25 Direct test mode (DTM) application.
28-Sep-2023	13	Added reference to STEVAL-IDB013V1, Section 31 Schematic diagrams, Section 36 Kit versions and Section 40 STEVAL-IDB013V1 regulatory compliance. Updated Section Introduction, Section 1.2 Kit contents, Section 2.3 STEVAL-IDB012V1 and STEVAL-IDB013V1 board overview, Section 2.5 BlueNRG-LPS SoC connections, Section 2.6 Power supply, Section 2.11 Jumpers, Section 2.12 Sensors, Section 2.13 Extension connector, Section 2.14 Push buttons, Section 2.15 LEDs, Section 2.16 CMSIS-DAP and Virtual COM, Section 2.16.3 USB_CMSISDAP programming/debugging feature, Section 2.17 BlueNRG-LP/BlueNRG-LPS programming and debugging, Section 2.18 Current measurements, Section 2.20 Hardware setup for STEVAL-IDB010V1/STEVAL-IDB012V1/STEVAL-IDB013V1 kit, Section 8 BLE Serial port demo application, Section 9 BLE Serial port master and slave demo application, Section 10 BLE remote control demo application, Section 11 BLE sensor profile demo, Section 12 BLE Sensor for STBLESensor app, Section 12.1 How to run the STBLESensor app for smartphones, Section 14 BLE HID/HOGP demonstration applications,

Date	Version	Changes
		Section 16 BLE notification consumer demonstration application, Section 17 BLE security demonstration applications, Section 18 BLE RC Long Range demonstration application, Section 21 BLE power consumption demo application, Section 22.1 Application roles, Section 23.1 Application roles, Section 24.1 Application roles.

Contents

List of figures

List of tables