

# Bluetooth<sup>®</sup> Low Energy Demo Applications User's Guide



# Contents

<b>Chapter 1 Introduction.....</b>	<b>6</b>
<b>Chapter 2 Bluetooth Low Energy.....</b>	<b>7</b>
<b>Chapter 3 Hardware Configurations.....</b>	<b>8</b>
3.1 Hardware requirements.....	8
3.2 Toolchain requirements.....	8
3.3 Freedom FRDM-KW41Z Platform.....	8
3.4 USB-KW41Z platform.....	9
3.5 QN9080CDK platform.....	9
3.6 Freedom FRDM-KW36 platform.....	10
3.7 Freedom FRDM-KW38 platform.....	11
3.8 USB-KW38 platform.....	12
<b>Chapter 4 Build and Run a Bluetooth LE Example Application.....</b>	<b>13</b>
4.1 User Interface.....	13
4.2 Security.....	13
4.3 Testing Devices.....	13
4.4 Time client devices.....	16
<b>Chapter 5 Bluetooth LE Stack and Demo Applications.....</b>	<b>17</b>
5.1 Heart rate sensor.....	17
5.1.1 Implemented Profile and Services.....	17
5.1.2 Supported platforms.....	17
5.1.3 User interface.....	17
5.1.4 Usage.....	18
5.2 Beacon.....	19
5.2.1 Supported platforms.....	19
5.2.2 Advertising data.....	20
5.2.3 User interface.....	20
5.2.4 Usage.....	20
5.2.5 Beacon usage with extended advertising.....	21
5.3 Blood pressure sensor.....	23
5.3.1 Implemented profile and services.....	23
5.3.2 Supported platforms.....	24
5.3.3 User interface.....	24
5.3.4 Usage.....	24
5.4 Glucose Sensor.....	26
5.4.1 Implemented profile and services.....	26
5.4.2 Supported platforms.....	26
5.4.3 User interface.....	26
5.4.4 Usage.....	26
5.5 Proximity Reporter.....	29
5.5.1 Implemented profile and services.....	29
5.5.2 Supported platforms.....	29
5.5.3 User interface.....	29
5.5.4 Usage.....	30

5.6 HID Device (Mouse).....	31
5.6.1 Implemented profiles and services.....	32
5.6.2 Supported platforms.....	32
5.6.3 User interface.....	32
5.6.4 Usage.....	32
5.7 HID Host.....	33
5.7.1 Implemented profiles and services.....	33
5.7.2 Supported platforms.....	34
5.7.3 User interface.....	34
5.7.4 Usage.....	34
5.8 Private Profile Server.....	35
5.8.1 Implemented profiles and services.....	35
5.8.2 Supported platforms.....	36
5.8.3 User interface.....	36
5.8.4 Usage.....	36
5.9 Private Profile Client.....	37
5.9.1 Implemented profiles and services.....	37
5.9.2 Supported platforms.....	37
5.9.3 User interface.....	37
5.9.4 Usage.....	38
5.10 Cycling Speed and Cadence Sensor.....	39
5.10.1 Implemented profiles and services.....	39
5.10.2 Supported platforms.....	39
5.10.3 User interface.....	39
5.10.4 Usage.....	39
5.11 Cycling Power Sensor.....	41
5.11.1 Implemented profiles and services.....	41
5.11.2 Supported platforms.....	41
5.11.3 User interface.....	41
5.11.4 Usage.....	41
5.12 Running Speed and Cadence Sensor.....	42
5.12.1 Implemented profiles and services.....	42
5.12.2 Supported platforms.....	42
5.12.3 User interface.....	42
5.12.4 Usage.....	43
5.13 Health Thermometer.....	44
5.13.1 Implemented profiles and services.....	45
5.13.2 Supported platforms.....	45
5.13.3 User interface.....	45
5.13.4 Usage.....	45
5.14 Low Power Temperature Sensor and Collector.....	47
5.14.1 Implemented profiles and services.....	47
5.14.2 Supported platforms.....	47
5.14.3 User interface.....	47
5.14.4 Usage.....	48
5.15 Low-Power Extended Advertising Peripheral and Extended Advertising Central (KW37 platforms).....	49
5.15.1 Implemented profile and services.....	49
5.15.2 Supported platforms.....	50
5.15.3 User interface.....	50
5.15.4 Usage.....	50
5.16 IPv6 Router and Node (KW41 Platforms).....	53
5.17 Alert Notification Server.....	53
5.17.1 Implemented profile and services.....	53
5.17.2 Supported platforms.....	53

5.17.3 User interface.....	53
5.17.4 Usage.....	54
5.18 Wireless UART.....	54
5.18.1 Implemented profile and services.....	54
5.18.2 Supported platforms.....	54
5.18.3 User interface.....	54
5.18.4 Usage.....	55
5.19 Bluetooth LE Shell.....	55
5.19.1 Implemented stack features.....	56
5.19.2 Implemented profile and services.....	56
5.19.3 Supported platforms.....	56
5.19.4 User interface.....	56
5.19.5 Usage.....	56
5.19.5.1 Extended Advertising (KW37 platforms).....	58
5.19.5.2 Periodic Advertising.....	59
5.19.5.3 RSSI Monitor.....	60
5.19.6 Throughput feature.....	62
5.20 Over the Air Programming (OTAP).....	63
5.20.1 Implemented profile and services.....	63
5.20.2 Supported platforms.....	63
5.20.3 User interface.....	63
5.20.4 Usage with Test Tool for Connectivity products.....	64
5.20.5 Usage with IoT Toolbox.....	70
5.21 Hybrid (Dual-Mode) Bluetooth Low Energy Heart Rate Sensor and IEEE 802.15.4 Coordinator Demo Application (KW41 Platform).....	72
5.21.1 Implemented profile and services.....	72
5.21.2 Supported platforms.....	72
5.21.3 User interface.....	73
5.21.4 Testing method.....	73
5.22 Relay Proxy.....	73
5.22.1 Implemented stack features.....	73
5.22.2 Implemented services.....	73
5.22.3 Supported platforms.....	74
5.22.4 User interface.....	74
5.22.5 Usage.....	74
5.23 Wireless Power Transfer System Receiving Unit and Transmitting Unit.....	74
5.23.1 Implemented profile and services.....	74
5.23.2 Supported platforms.....	75
5.23.3 User interface.....	75
5.23.4 Usage.....	76
5.24 Hybrid (Dual-Mode) Bluetooth Low Energy and Generic FSK Demo Application.....	78
5.24.1 Implementation.....	78
5.24.1.1 How it works.....	79
5.24.2 Supported platforms.....	80
5.24.3 User interface.....	80
5.24.4 Customization.....	80
5.24.5 Wireless usage with extended advertising.....	81
5.25 ANCS Client.....	82
5.25.1 Implemented profile and services.....	82
5.25.2 Supported platforms.....	82
5.25.3 User interface.....	82
5.25.4 Usage.....	83
5.26 Bluetooth LE FSCI Black Box.....	85
5.26.1 Description.....	85
5.26.2 Supported platforms.....	85

5.26.3 Usage with Test Tool for Connectivity Products.....	85
5.27 HCI Black Box.....	87
5.27.1 Description.....	87
5.27.2 Supported platforms.....	87
5.27.3 Usage with Test Tool for Connectivity Products.....	88
5.27.4 Direct Test Mode.....	89
5.28 Pulse Oximeter Sensor.....	90
5.28.1 Implemented profile and services.....	90
5.28.2 Supported platforms.....	90
5.28.3 User interface.....	90
5.28.4 Usage.....	90
5.29 Location and Navigation Server and Client.....	90
5.29.1 Implemented profile and services.....	91
5.29.2 Supported platforms.....	91
5.29.3 User interface.....	91
5.29.4 Usage.....	91
<b>Chapter 6 Revision history.....</b>	<b>93</b>

# Chapter 1

## Introduction

This document describes the Bluetooth® Low Energy stack enablement for the NXP development platforms. The initial chapters start with a presentation of the hardware and toolchain requirements.

Chapter 5 describes the demo applications that can be found in the software development kit. It presents the profiles and services implemented and how to interact with them.

# Chapter 2

## Bluetooth Low Energy

The Software Development Package provides a Bluetooth Low Energy v5.0-compliant host stack and controller implementation with a set of GATT-based profiles and services implemented on top. To demonstrate the device functionality, the following demo applications are implemented.

- Bluetooth LE Relay Proxy
- Bluetooth LE Shell App
- IPv6 Node and Router (KW41 platforms)
- OTAP Client and Server
- HID Device (Mouse) and HID Host
- Low Power Temperature Sensor and Collector
- Wireless UART demo application
- Proximity Reporter
- Alert Notification Server
- Heart Rate Sensor
- Blood Pressure Sensor
- Glucose Sensor
- Health Thermometer
- Cycling Speed and Cadence Sensor
- Cycling Power Sensor
- Running Speed and Cadence Sensor
- Beacon Application
- Wireless Power Transfer Receiving and Transmitting Units (KW41 and KW3x Platforms)
- Private Profile Client and Server
- ANCS Client
- Bluetooth LE FSCI Black Box
- HCI Black Box
- Pulse Oximeter Sensor
- Location and Navigation Server
- Location and Navigation Client
- Hybrid Bluetooth LE and Generic FSK advertising demo application

# Chapter 3

## Hardware Configurations

### 3.1 Hardware requirements

The NXP Bluetooth LE demo applications are designed to run on any of these supported platforms:

- FRDM-KW41Z
- USB-KW41Z
- QN908XCDK
- FRDM-KW36
- FRDM-KW38
- USB-KW38

### 3.2 Toolchain requirements

The Bluetooth Low Energy Stack demo applications were compiled and tested with IAR Embedded Workbench for Arm<sup>®</sup> and MCUXpresso. It is recommended to use one of these tools.

### 3.3 Freedom FRDM-KW41Z Platform

FRDM-KW41Z Freedom platform is based on the KW41Z wireless, dual-mode SoC, which incorporates an Arm<sup>®</sup> Cortex<sup>®</sup>-M0+ core and can be configured to operate at various frequencies, up to 48 MHz. It has 512 KB of Flash and 128 KB of SRAM. A figure representing the platform is shown below. For detailed information about the platform, see the appropriate board user's guide.

The platform features a composite USB device called OpenSDA which serves as debugger interface and as a USB-to-serial converter via a virtual COM port application. Several firmware images can be programmed on the OpenSDA device, such as:

- [developer.mbed.org/handbook/CMSIS-DAP](https://developer.mbed.org/handbook/CMSIS-DAP)
- [segger.com/opensda.html](https://segger.com/opensda.html)
- [www.pemicro.com/opensda](https://www.pemicro.com/opensda)



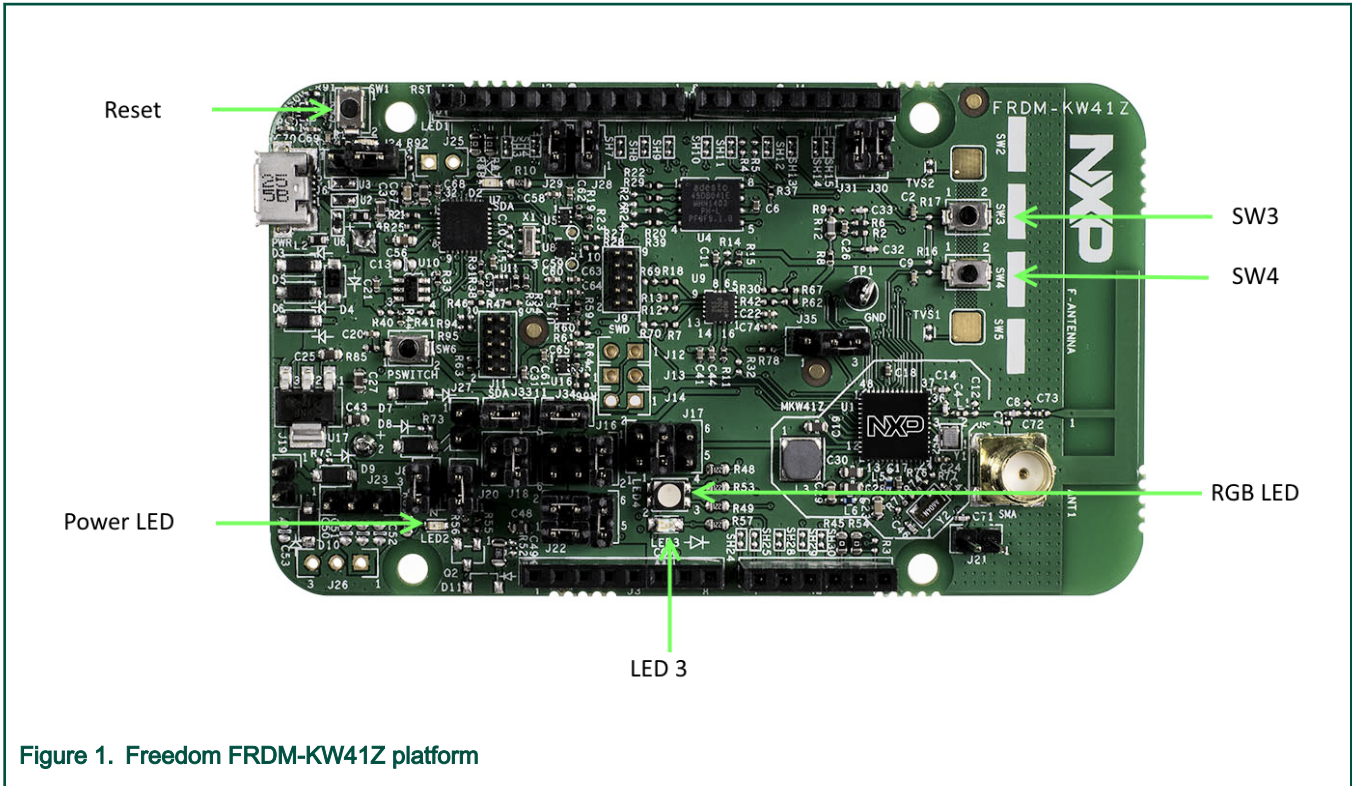


Figure 1. Freedom FRDM-KW41Z platform

### 3.4 USB-KW41Z platform

The USB-KW41Z board is mainly targeted for sniffer applications. It is based on the same KW41Z wireless, dual mode SoC. A figure representing the platform is shown below. For detailed information about the board, see the appropriate board user's guide.

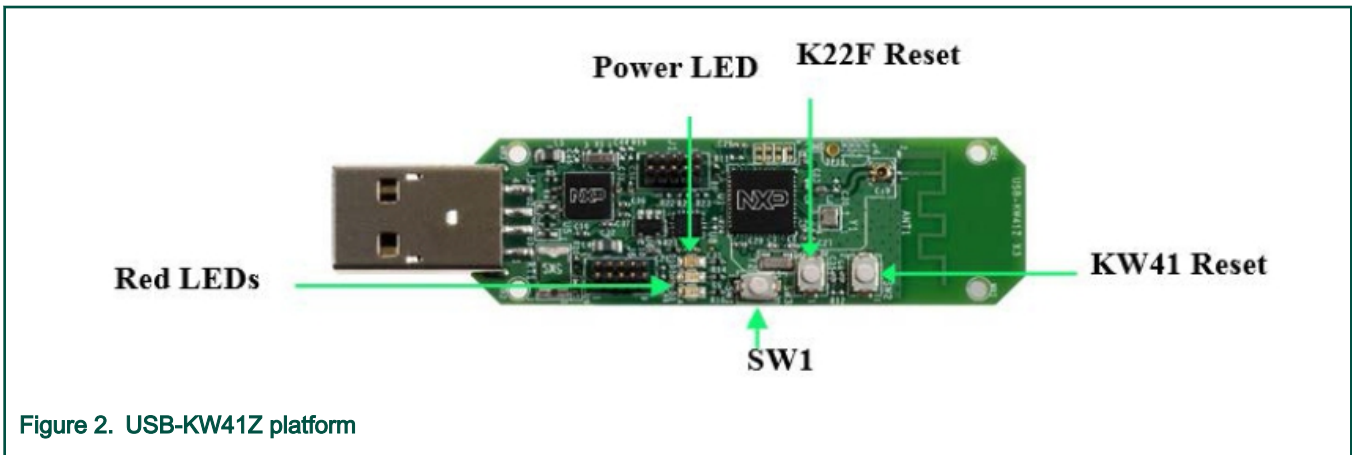


Figure 2. USB-KW41Z platform

### 3.5 QN9080CDK platform

QN9080CDK platform is based on the QN9080CDK Bluetooth Low Power SoC, which incorporates an Arm® Cortex®-M4F core and can be configured to operate at various frequencies, up to 32 MHz. It has 512 KB of Flash and 128 KB of SRAM. A figure representing the platform is shown below. For detailed information about the platform, see the appropriate board user's guide.

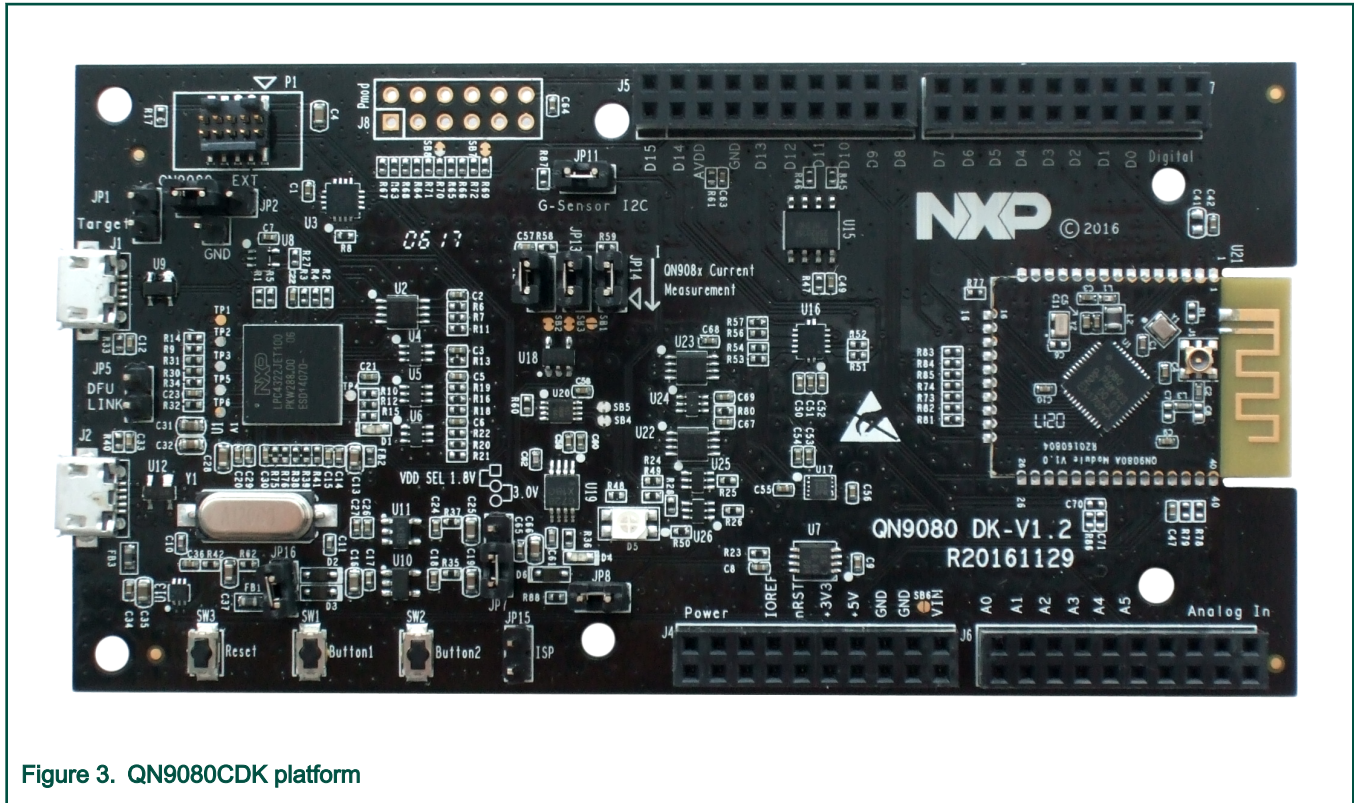


Figure 3. QN9080CDK platform

### 3.6 Freedom FRDM-KW36 platform

FRDM-KW36 Freedom platform is based on the KW36 SoC which incorporates an Arm<sup>®</sup> Cortex<sup>®</sup>-M0+ core and enables Bluetooth Low Energy and Generic FSK. It can be configured to operate at various frequencies, up to 48 MHz and it has 256 KB of on-chip Flash and 64 KB of SRAM. The Bluetooth LE transceiver supports up to 8 concurrent connections in any master/slave combination. A figure representing the platform is shown below. For detailed information about the platform, see the appropriate board user's guide.

The platform features a composite USB device called OpenSDA which serves as debugger interface and as a USB-to-serial converter via a virtual COM port application.

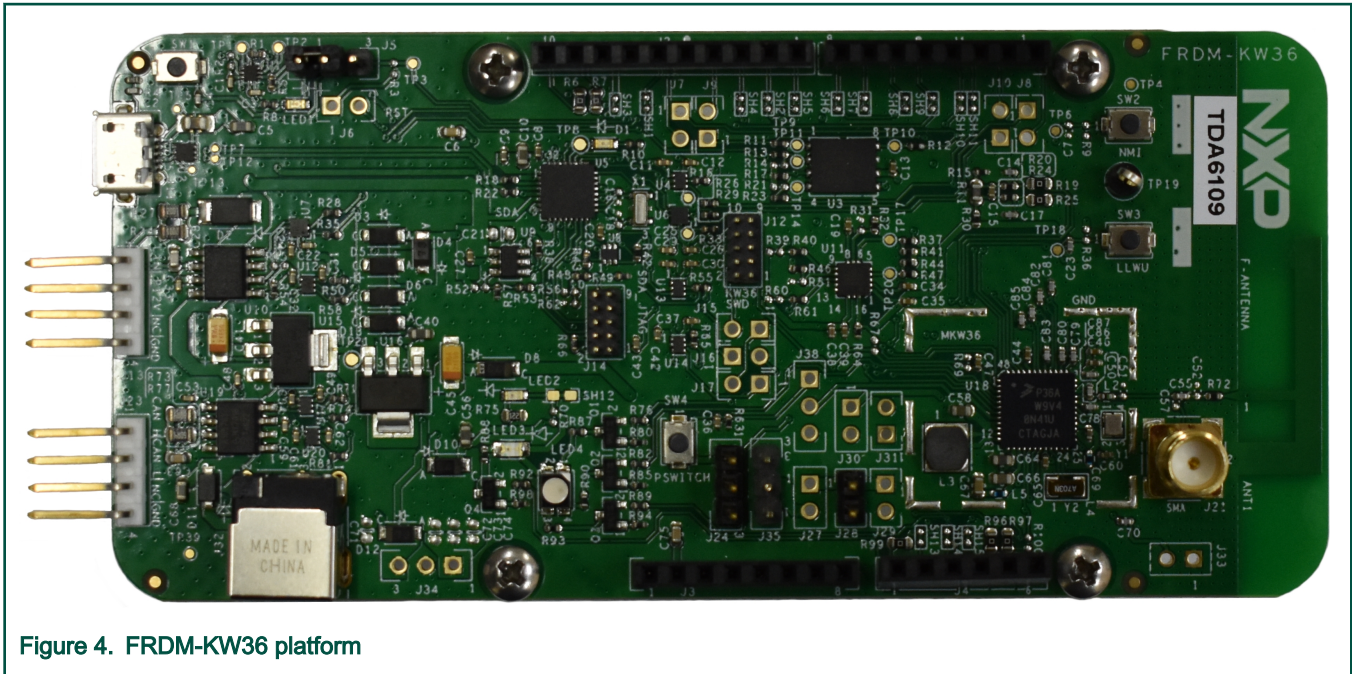


Figure 4. FRDM-KW36 platform

### 3.7 Freedom FRDM-KW38 platform

The FRDM-KW38 board is based on the KW38A wireless SoC that integrates an Arm® Cortex®-M0+ CPU running at frequencies up to 48 MHz, up to 512 KB FLASH memory and 64 KB SRAM and a 2.4 GHz radio that supports Bluetooth LE 5.0 and GenFSK modulations. The chip is intended for automotive and industrial applications, being fully AEC Q100 Grade 2 Automotive Qualified.

Target applications are Digital Key, BLE Key Fob (that can replace traditional sub-1GHz) key fobs, sensors and localization.

The KW38 wireless MCU also includes a FlexCAN module enabling integration into a vehicle or industrial CAN communication network. The FRDM-KW38 board features a composite USB device called OpenSDA which serves as debugger interface and as a USB to serial converter via a virtual COM port application. For detailed information about the platform, see the appropriate board user's guide.

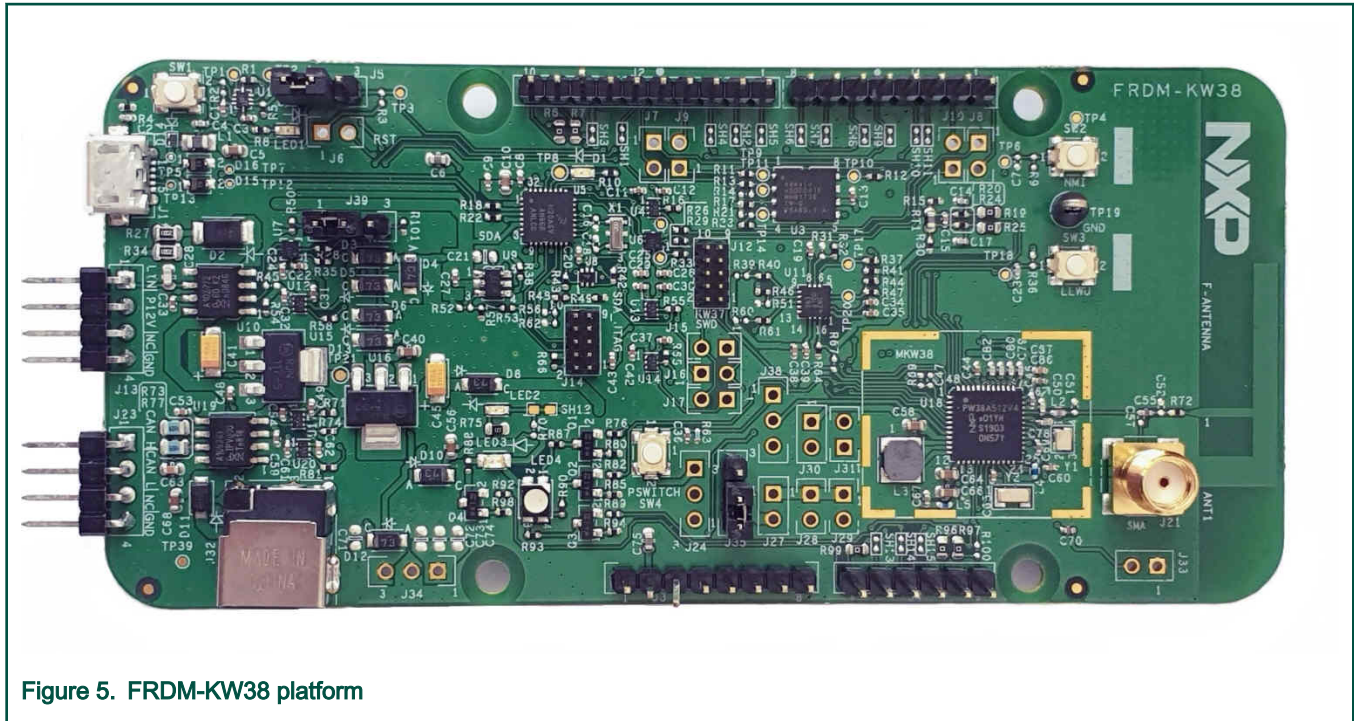


Figure 5. FRDM-KW38 platform

### 3.8 USB-KW38 platform

The USB-KW38 board is mainly targeted for sniffer applications. It is based on the KW37Z SoC. A figure representing the platform is shown below. For detailed information about the board, see the appropriate board user's guide.

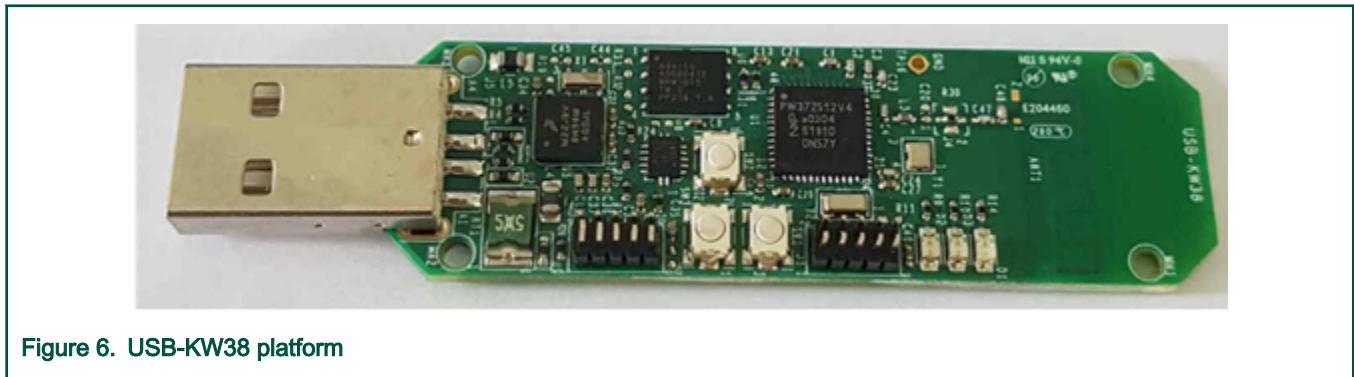


Figure 6. USB-KW38 platform

# Chapter 4

## Build and Run a Bluetooth LE Example Application

To open, build and run any example application, see the corresponding board Bluetooth Low Energy Quick Start Guide document.

### 4.1 User Interface

The demo applications that implement the Battery Service expose the current battery level, as measured on the board, through the Battery Level characteristic. The value represents a percentage between 0 and 100. The value can be read from the device from a connected GATT client.

The demo applications that implement the Device Information Service expose different information regarding the current software, hardware, and firmware revisions. These values are used as an example and can be modified by the user when developing their product. The values can be read from the device from a connected GATT client.

### 4.2 Security

The examples that enable pairing always generates a default passkey of 999999 that has to be entered on the central device, in most cases a smartphone or tablet.

### 4.3 Testing Devices

To demonstrate the profile functionality, most of the scenarios require one of the supported platform and a Bluetooth Low Energy capable central device, usually a smartphone or a tablet that runs a compatible Bluetooth LE application.

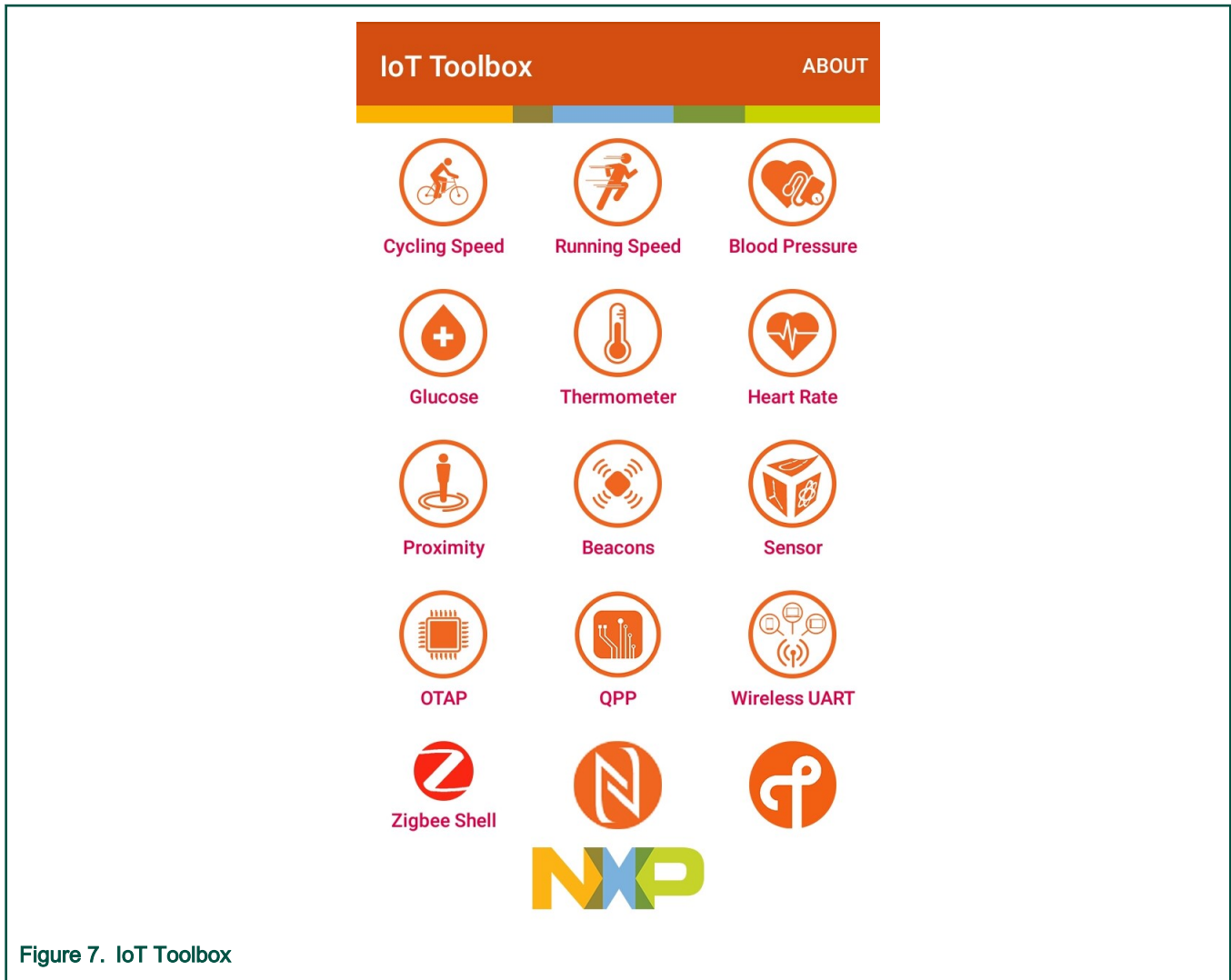


Figure 7. IoT Toolbox

The recommended application is the IoT Toolbox, which can be installed on Apple® iOS or Android™ OS handheld devices that support Bluetooth Low Energy. The application can be found on [iTunes](#) or [Google playstore](#).

Other demos can be run by using two platforms, one for the peripheral and one for the central role, for example IPv6 Node and Router, Low Power Temperature Sensor and Collector, Wireless UART, OTAP Client and Server, HID Host and Device.

To provide feedback and additional interaction, some examples make use of a shell console via the virtual COM port. To access the device, open a serial port terminal and connect it to the platform as shown in the figures below. For this example, Tera Term VT and a FRDM-KW41Z were used. The communication parameters are 115200 and 8N1.

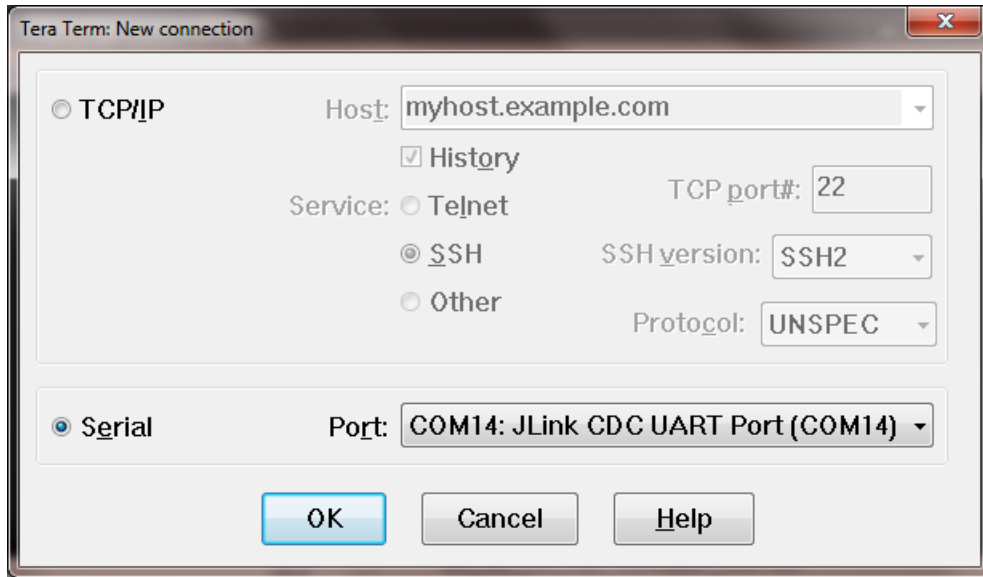


Figure 8. Tera Term – mbed serial port

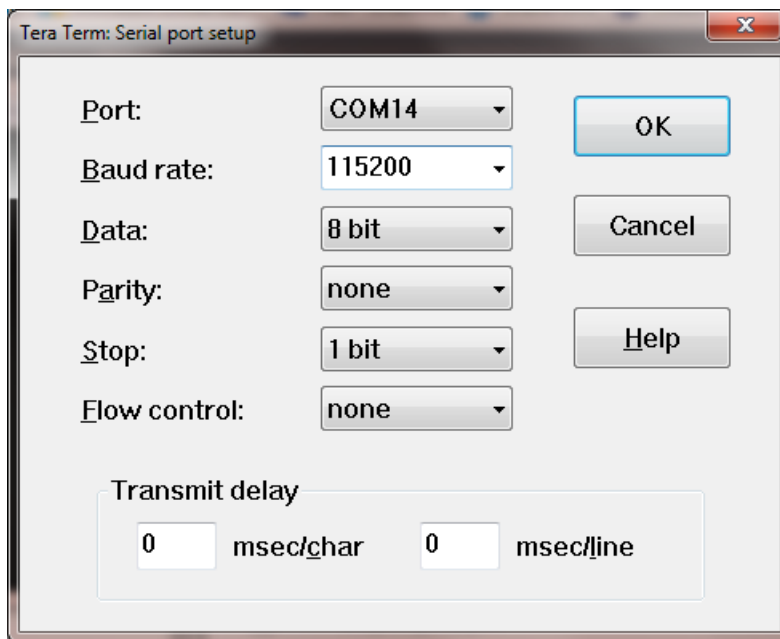


Figure 9. Tera Term – mbed serial port configuration

The start screen is displayed after the board is reset.

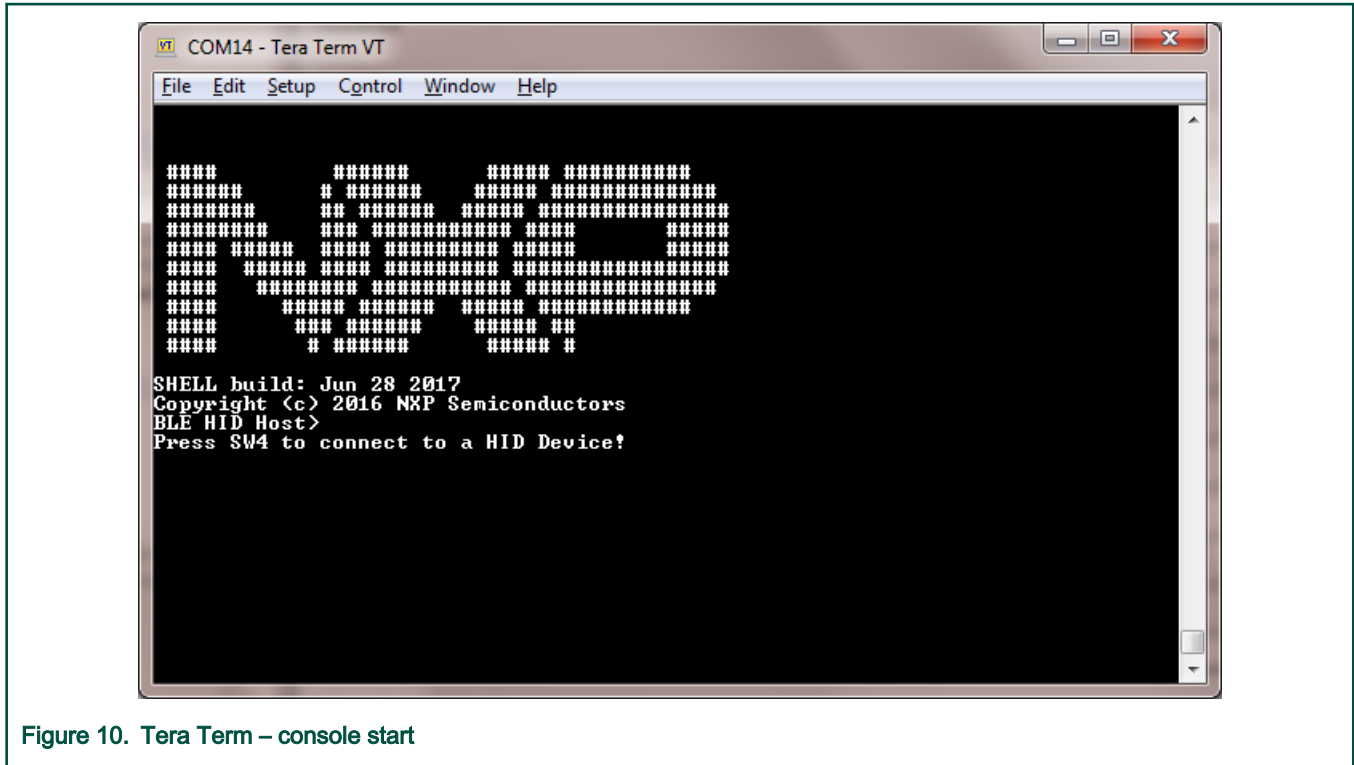


Figure 10. Tera Term – console start

### 4.4 Time client devices

Some applications implement the Current Time Service. To enable this feature, define `gAppUseTimeService_d` in the `app_preinclude.h` file as 1. If the Time Client is enabled, the device must synchronize with a Time Server to update its internal date/time to the current date/time. The Time Client synchronizes with a phone if you connect the device to it (pairing and bonding must be active).



# Chapter 5

## Bluetooth LE Stack and Demo Applications

### 5.1 Heart rate sensor

This section describes the implemented profiles and services, user interactions, and testing methods for the Heart Rate Sensor application.

#### 5.1.1 Implemented Profile and Services

The Heart Rate Sensor application implements a GATT server and the following profile and services.

- Heart Rate Profile v1.0
- Heart Rate Service v1.0
- Battery Service v1.0
- Device Information Service v1.1

The application behaves as a GAP peripheral node. It enters the GAP General Discoverable Mode and waits for a GAP central node to connect. When the collector configures notifications, the sensor starts sending heart rate measurements every second.

#### 5.1.2 Supported platforms

The Heart Rate Sensor application is supported by the following platforms:

- FRDM-KW41Z
- USB-KW41Z
- QN9080CDK
- FRDM-KW36
- FRDM-KW38

#### 5.1.3 User interface

After flashing the board, the sensor is put in deep sleep (all LEDs are off). Also, any attached debugger will lose its connection. To flash the board in case the sensor is put in deep sleep, press ADVSW or RESET. The application default configuration enables low power, which disables LED support. The user can manually change this and enable LED support, otherwise all subsequent LED behavior references shall be ignored. To wake up and start advertising, press the ADVSW button. When in GAP Discoverable Mode, the CONNLED is on. When the central node connects to the peripheral, the CONNLED remains solid. The heart rate measurement values are generated randomly in the interval 40 to 200 bpm. The heart rate value format is set to 8 bits and the contact status is on. For FRDM-KW38 board, the default enabled low-power modes are 1 and 5. This means that the MCU is put into VLLS2 or VLLS3 stop mode. In this mode, the LED module operation is not implemented.

See details below for hardware references.

**Table 1. Hardware references**

Platform	ADVSW	CONNLED
FRDM-KW41Z	SW4	LED4
QN9080CDK	BUTTON1	D5-RED

*Table continues on the next page...*

**Table 1. Hardware references (continued)**

Platform	ADVSW	CONNLED
FRDM-KW36	SW3	LED3
FRDM-KW38	SW3	LED3

### 5.1.4 Usage

The Heart Rate Sensor can be connected to any Bluetooth® Smart Ready products available on the market.

To make the Heart Rate Sensor visible, press the ADVSW button to start sending advertisements. The sensor name “NXP\_HRS” shows on the device when its scanning is active. If configured, the sensor notifies the application with heart rate measurements every second. Also, the battery level and various device information is exposed for reading. The IoT Toolbox can be used to showcase the profile functionality, as shown in the image below.

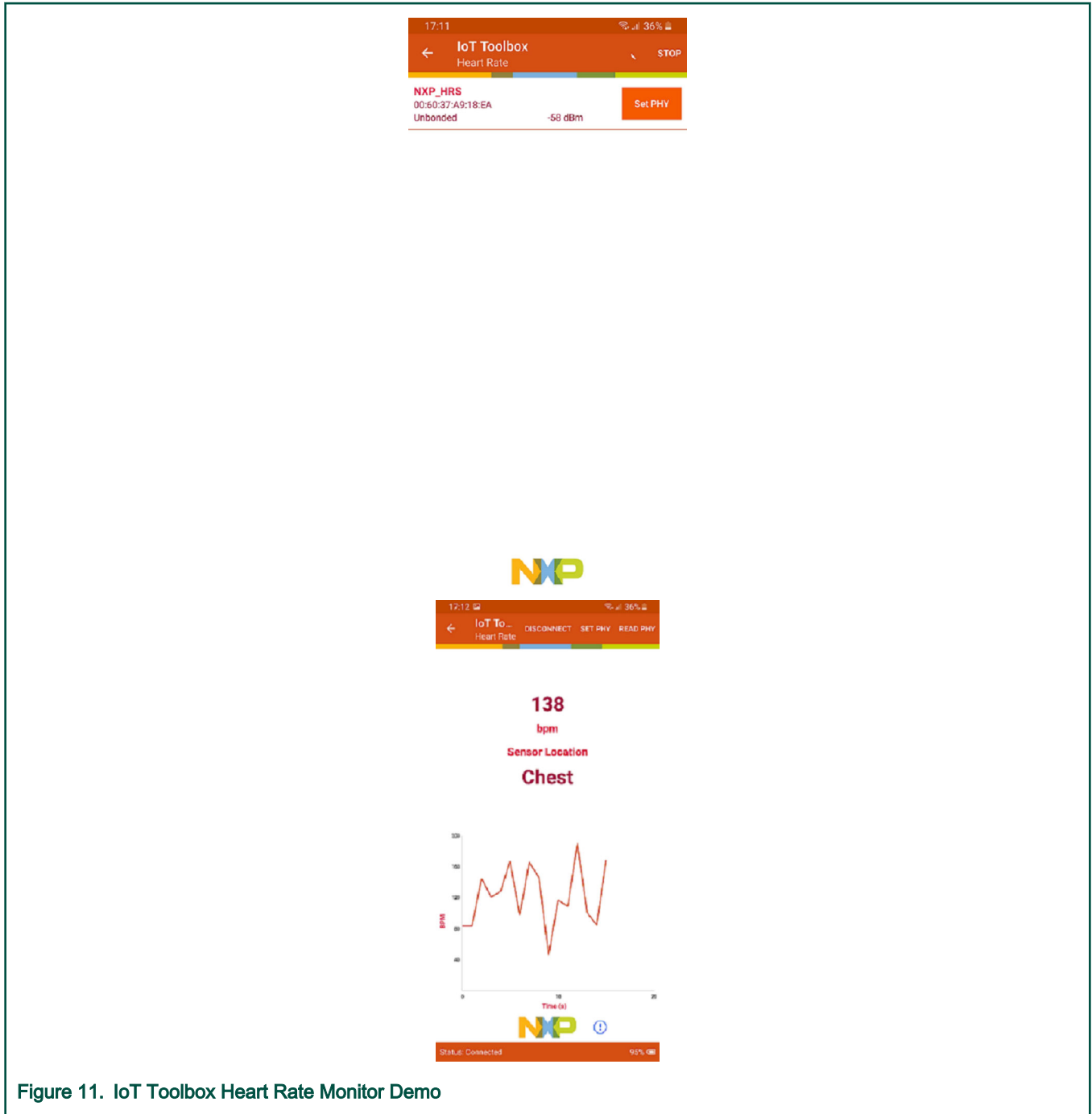


Figure 11. IoT Toolbox Heart Rate Monitor Demo

## 5.2 Beacon

This section presents the user interactions and testing methods for the Beacon application.

### 5.2.1 Supported platforms

The Beacon application is supported by the following platforms:

- FRDM-KW41Z
- USB-KW41Z

- QN9080CDK
- FRDM-KW36
- FRDM-KW38

## 5.2.2 Advertising data

The beacons are non-connectable advertising packets that are sent on the three advertising channels. The latter contains the following fields.

- Company Identifier (2 bytes): 0x0025 (NXP ID as defined by the Bluetooth SIG).
- Beacon Identifier (1 byte): 0xBC (Allows identifying an NXP Beacon alongside with Company Identifier).
- UUID (16 bytes): Beacon sensor unique identifier.
- A (2 bytes): Beacon application data.
- B (2 bytes): Beacon application data.
- C (2 bytes): Beacon application data.
- RSSI at 1m (1 byte): Allows distance-based applications.

By default, the UUID value is a random value based on the unique identifier of the board.

## 5.2.3 User interface

After flashing the board, the sensor is put in deep sleep (all LEDs are off). To flash the board in case the sensor is put in deep sleep, press ADVSW or RESET. Also, any attached debugger loses its connection. The application default configuration enables low power, which disables LED support. The user can manually change the configuration and enable the LED support, otherwise all subsequent LED behavior references shall be ignored. First press of the ADVSW button starts legacy advertising. The following applies to platforms capable of performing Extended Advertising - which excludes QN9090/K32W061/KW35. Second press starts extended advertising, and the third press starts periodic advertising. Another press stops the periodic advertising, and another one stops the extended advertising as well. For FRDM-KW38 board, the default enabled low-power modes are 1 and 5. This means that the MCU is put into VLLS2 or VLLS3 stop mode. In this mode, the LED module operation is not implemented.

See details below for hardware references.

**Table 2. Hardware references**

Platform	ADVSW	CONNLED
FRDM-KW41Z	SW4	LED3
QN9080CDK	BUTTON1	D5-RED
FRDM-KW36	SW2	LED3
FRDM-KW38	SW3	LED3

## 5.2.4 Usage

The beacon can be tested with any Bluetooth<sup>®</sup> Smart Ready products available on the market. The IoT Toolbox can also be used to showcase the profile functionality, as shown in the image below.

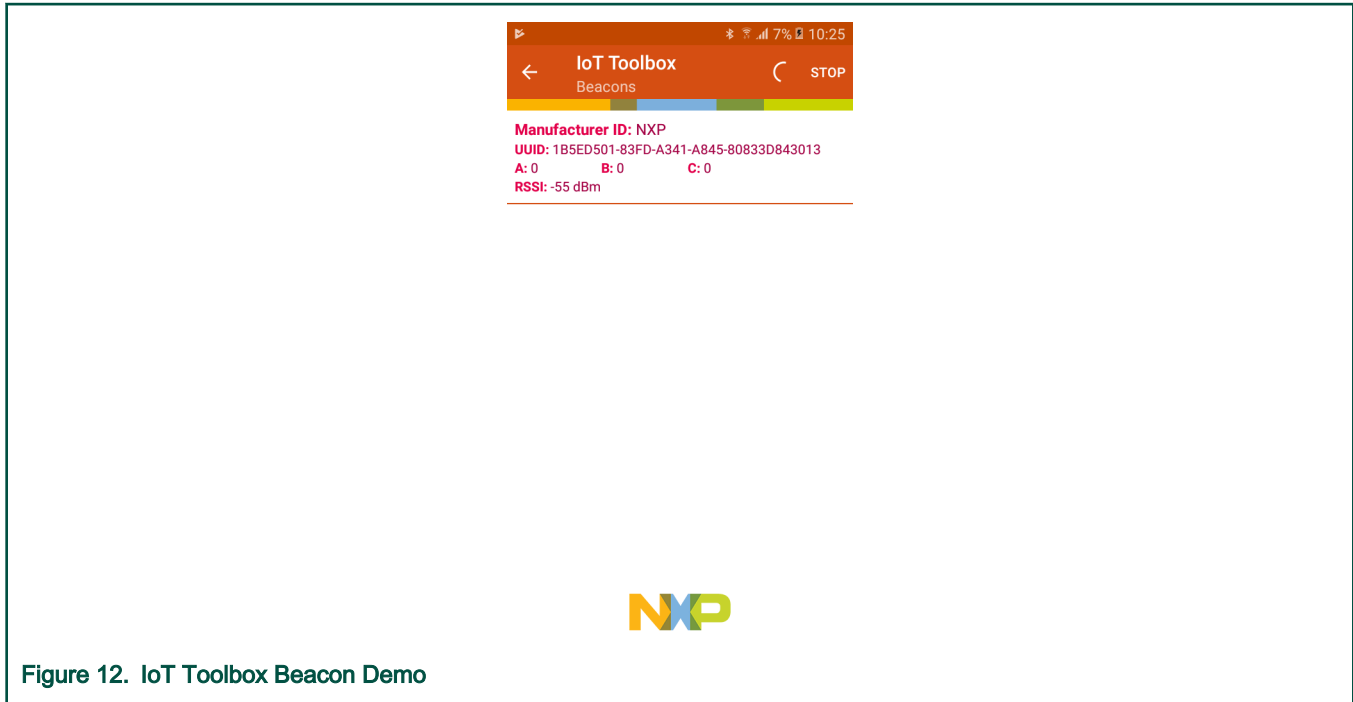


Figure 12. IoT Toolbox Beacon Demo

### 5.2.5 Beacon usage with extended advertising

To use the Beacon application with the advertising extensions capabilities, the `gBeaconAE_c` define must be set to 1. Doing this enables the usage of extended advertising and periodic advertising. The application cycles between these modes are in the following manner (the EXTADVSW button for FRDM-KW38 is SW3):

1. First press EXTADVSW button starts legacy advertising
2. Second press EXTADVSW starts extended advertising on the coded phy, while also keeping the legacy advertising active.
3. Third press EXTADVSW starts periodic advertising on the coded phy, while also keeping the legacy and the extended advertising previously started active.
4. Fourth press EXTADVSW stops periodic advertising. Legacy and extended advertising remain active.
5. Fifth press EXTADVSW stops extended advertising. Legacy advertising remains active.
6. Sixth press EXTADVSW stops legacy advertising

As not all smartphones support extended advertising, a different method to view the AE beacon is to use the `ble_shell` application. In order to do this, the following steps must be taken:

1. Flash a board with the beacon application, as described above.
2. Flash a board with the `ble_shell` application, as described in [Bluetooth LE Shell](#) and connect to it using a serial port.
3. Press the EXTADVSW button two times on the beacon to start extended advertising on the coded phy.
4. To view the advertising data, enter the following commands in the shell terminal to set the scanning phy to coded and start scanning.

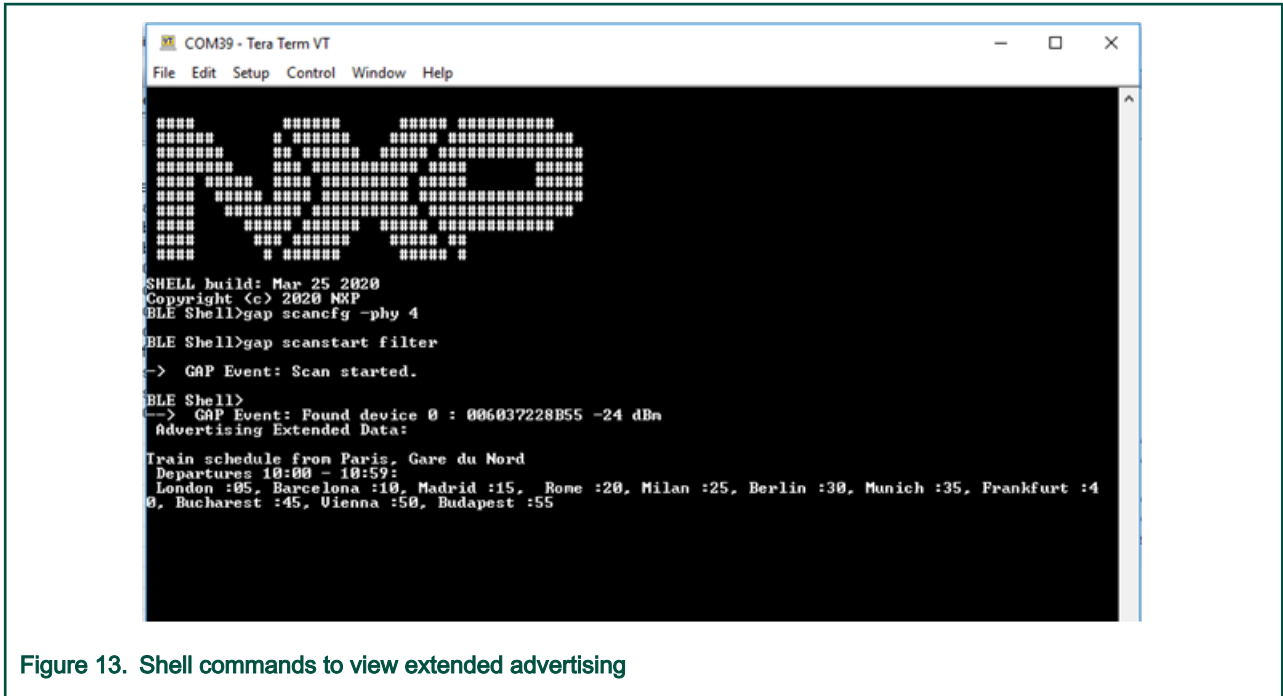


Figure 13. Shell commands to view extended advertising

5. To start the periodic advertising, press EXTADVSW button again on the beacon.
6. To sync with the beacon, issue the following commands on the shell terminal:

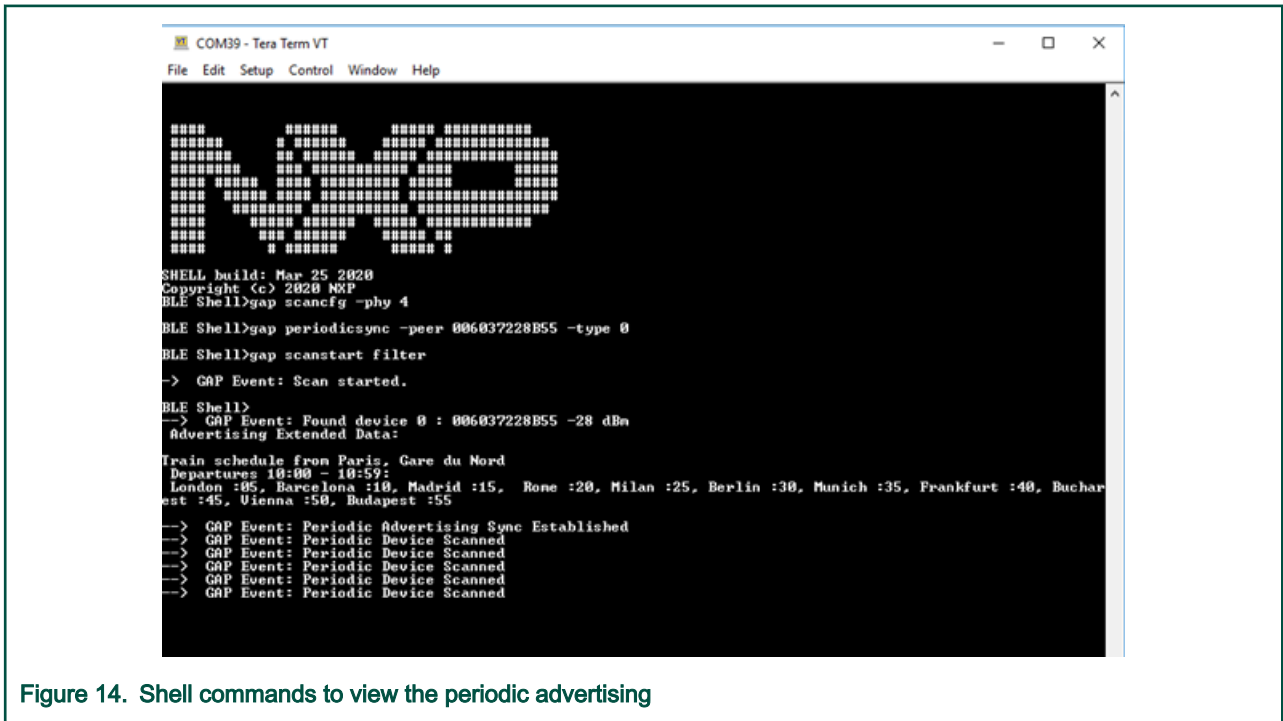


Figure 14. Shell commands to view the periodic advertising

As one can see, the extended advertising is still on when periodic advertising is started.

The peer parameter of the periodicsync command is the public address of the beacon.

### Extended Advertising with very large data

To use very large advertising data for extended advertising, set the gBeaconLargeExtAdvData\_c define to 1. The same steps are used to view the data using ble\_shell:

1. Flash a board with the beacon application
2. Flash a board with the ble\_shell application, as described in [Bluetooth LE Shell](#) and connect to it using a serial port.
3. Press the EXTADVSW button two times on the beacon to start extended advertising on the coded phy.
4. To view the advertising data, enter the following commands in the shell terminal to set the scanning phy to coded and start scanning.

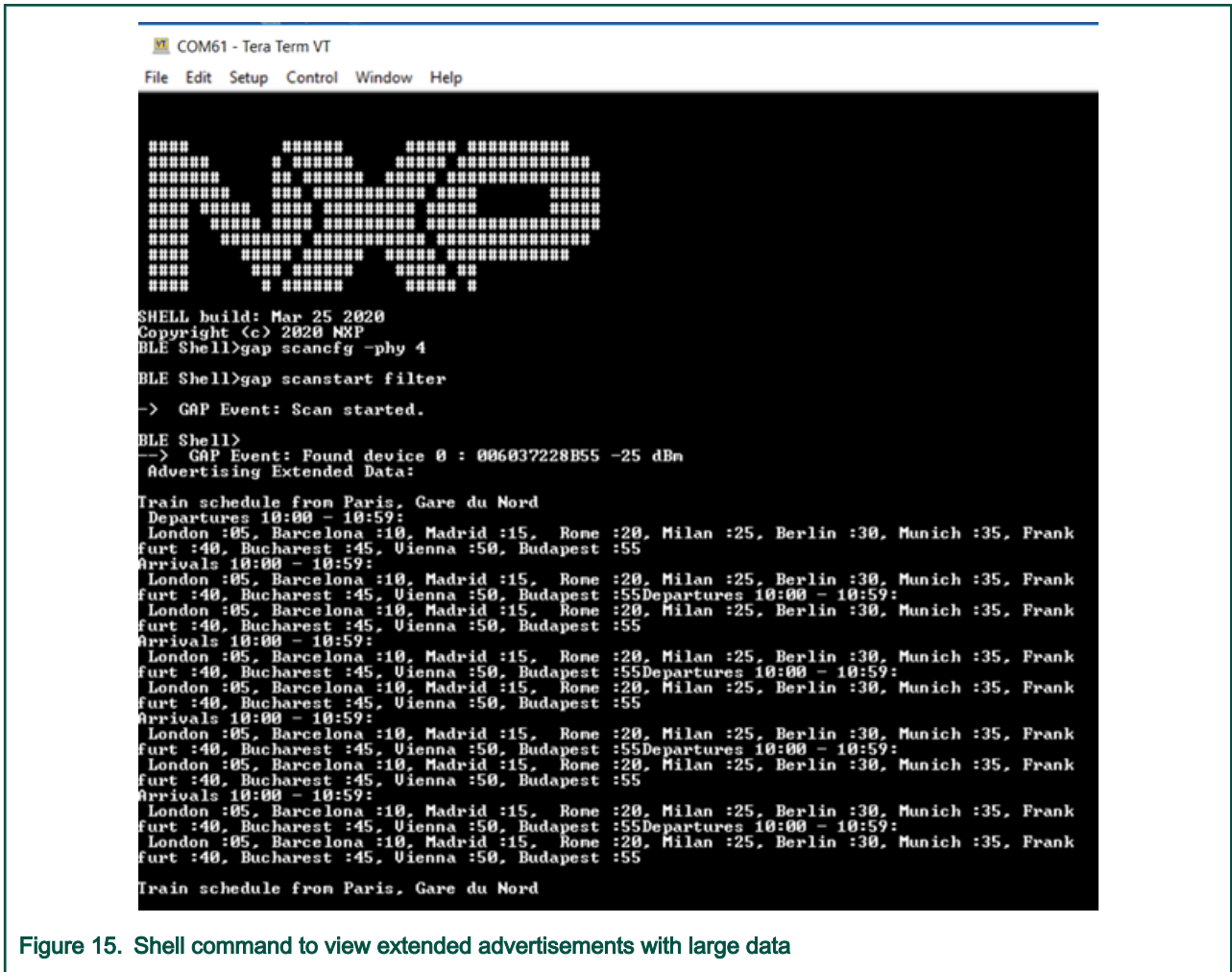


Figure 15. Shell command to view extended advertisements with large data

### 5.3 Blood pressure sensor

This section describes implemented profiles and services, user interactions, and testing methods for the Blood Pressure Sensor application.

#### 5.3.1 Implemented profile and services

The Blood Pressure Sensor application implements a GATT server and the following profile and services.

- Blood Pressure Profile v1.0
- Blood Pressure Service v1.0
- Battery Service v1.0
- Device Information Service v1.1
- Current Time Service v1.1

The application behaves as a GAP peripheral node. It enters GAP General Discoverable Mode and waits for a GAP central node to connect. When the collector configures notifications or indications, the sensor starts sending intermediate cuff pressure and blood pressure measurements. In a 5 second interval, the sensor sends 5 events (4 notifications with the intermediate cuff pressure measurement and one indication with the blood pressure measurement).

### 5.3.2 Supported platforms

The Blood Pressure Sensor application is supported by the following platforms:

- FRDM-KW41Z
- USB-KW41Z
- QN9080CDK
- FRDM-KW36
- FRDM-KW38

### 5.3.3 User interface

After flashing the board, the sensor is in idle mode (all LEDs flashing). To start advertising, press the ADVSW button.

When in GAP Discoverable Mode, CONNLED is flashing. When the central node connects to the peripheral, CONNLED turns solid. To disconnect the node, hold the ADVSW button pressed for 2-3 seconds. The node then re-enters GAP Discoverable Mode.

The blood and cuff pressure measurement values are generated randomly on the device.

See details below for hardware references.

**Table 3. Hardware references**

Platform	ADVSW	CONNLED
FRDM-KW41Z	SW4	LED3
QN9080CDK	BUTTON1	D5-RED
FRDM-KW36	SW2	LED3
FRDM-KW38	SW2	LED3

### 5.3.4 Usage

The blood pressure sensor can be connected to any Bluetooth® Smart Ready products available on the market.

To make the sensor visible, press the ADVSW button to start sending advertisements, which causes CONNLED to start flashing. The sensor name “NXP\_BPS” shows on the device when its scanning is active. A solid CONNLED indicates a successful connection between the 2 devices. Hold the ADVSW button pressed for 2-3 seconds anytime to initiate disconnect.

If configured, the sensor notifies the application with intermediate cuff pressure each second and blood pressure measurements every 5 seconds. Also, the battery level and various device information is exposed for reading. The IoT Toolbox can be used to showcase the profile functionality, as shown in the image below.



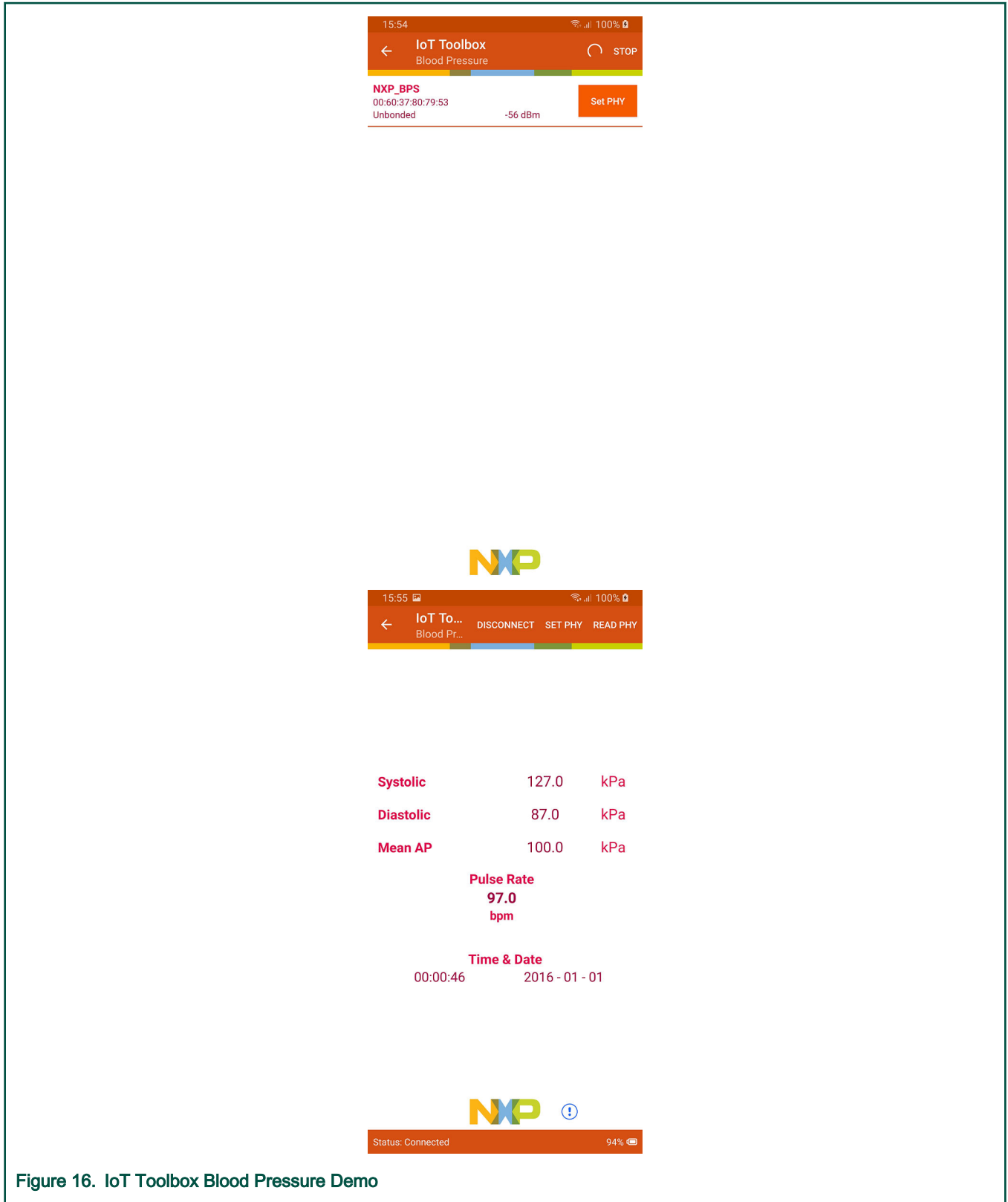


Figure 16. IoT Toolbox Blood Pressure Demo

## 5.4 Glucose Sensor

This section describes implemented profiles and services, user interactions, and testing methods for the Glucose Sensor application.

### 5.4.1 Implemented profile and services

The Glucose Sensor application implements a GATT server and the following profile and services.

- Glucose Profile v1.0
- Glucose Service v1.0
- Battery Service v1.0
- Device Information Service v1.1
- Current Time Service v1.1

The application behaves as a GAP peripheral node. It enters GAP General Discoverable Mode and waits for a GAP central node to connect. When the collector configures notifications, the sensor can send glucose measurements. The latest three measurements are stored on the sensor device and can be retrieved by the collector.

### 5.4.2 Supported platforms

The Glucose Sensor application is supported by the following platforms:

- FRDM-KW41Z
- USB-KW41Z
- QN9080CDK
- FRDM-KW36
- FRDM-KW38

### 5.4.3 User interface

After flashing the board, the sensor is in idle mode (all LEDs flashing). To start advertising, press the ADVSW button. When in GAP Discoverable Mode, CONNLED is flashing. When the central node connects to the peripheral, CONNLED turns solid. To disconnect the node, hold the ADVSW button pressed for 2-3 seconds. The node then re-enters GAP Discoverable Mode. When in connection, pressing the ADVSW button triggers a new glucose measurement. The glucose measurement values are generated randomly on the device.

See details below for hardware references.

**Table 4. Hardware references**

Platform	ADVSW	CONNLED
FRDM-KW41Z	SW4	LED3
QN9080CDK	BUTTON1	D5-RED
FRDM-KW36	SW2	LED3
FRDM-KW38	SW2	LED3

### 5.4.4 Usage

The glucose sensor can be connected to any Bluetooth® Smart Ready products available on the market.

To make the sensor visible, press the ADVSW button to start sending advertisements, which causes CONNLED to start flashing. The sensor name "NXP\_GLS" shows on the device when its scanning is active. A solid CONNLED indicates a successful connection between the two devices. Hold the ADVSW button pressed for 2-3 seconds anytime to initiate disconnect.

If notifications are configured, the sensor sends the data to the collector when ADVSW button is pressed for less than 1 second. The measurement data is stored on the sensor. Also, the battery level and various device information is displayed for reading. The IoT Toolbox can be used to showcase the profile functionality, as shown in the image below.



Figure 17. IoT Toolbox Glucose Sensor Demo

## 5.5 Proximity Reporter

This section describes the implemented profiles and services, user interactions and testing methods for the Proximity Reporter application.

### 5.5.1 Implemented profile and services

The Proximity Reporter application implements a GATT server and the following profile and services.

- Proximity Profile v1.0.1
- Immediate Alert Service v1.0
- TX Power Service v1.0
- Link Loss Service v1.0.1
- Battery Service v1.0
- Device Information Service v1.1

The application behaves as a GAP peripheral node. It enters GAP General Discoverable Mode and waits for a GAP central node to connect.

The Immediate Alert Service shows the alert level control point. GATT clients can trigger an alert on the device. The TX Power Service exposes the TX Power level that is read from the Bluetooth LE controller. The Link Loss Service exposes a configurable alert level. When the radio link with the central node is suddenly lost, the configured alert is triggered on the reporter node.

### 5.5.2 Supported platforms

The Proximity Reporter application is supported by the following platforms:

- FRDM-KW41Z
- USB-KW41Z
- QN9080CDK
- FRDM-KW36
- FRDM-KW38

### 5.5.3 User interface

After flashing the board, the sensor is in idle mode (all LEDs flashing). To start advertising, press the ADVSW button. When in GAP Discoverable Mode, CONNLED is flashing. When the central node connects to the peripheral, CONNLED turns solid. To disconnect the node, hold the ADVSW button pressed for 2-3 seconds. The node then re-enters GAP Discoverable Mode.

Alerts are notified using an LED. On platforms other than QN9090/K32W061 DK6, the RGB LED is used. When a mild alert is triggered on the reporter, RGB LED starts flashing blue. When a high alert is triggered on the reporter, both RGB LED starts flashing green. On QN9090/K32W061 DK6, LED DS3 is used for that purpose. Being a plain red LED, it is turned-off in the absence of alert, it turns red solid for 'mild' alerts and starts flashing on reception of 'high' alerts.

See details below for hardware references.

**Table 5. Hardware references**

Platform	ADVSW	CONNLED
FRDM-KW41Z	SW4	LED3
QN9080CDK	BUTTON1	D5-RED

*Table continues on the next page...*

**Table 5. Hardware references (continued)**

Platform	ADVSW	CONNLED
FRDM-KW36	SW2	LED3
FRDM-KW38	SW2	LED3

### 5.5.4 Usage

The proximity reporter can be connected to any Bluetooth® Smart Ready products available on the market. To make the sensor visible, press ADVSW to start advertising which causes CONNLED to start flashing. The sensor name “NXP\_PXR” shows on the device when its scanning is active. A solid CONNLED indicates a successful connection between the two devices. Hold the ADVSW button pressed for 2-3 seconds anytime to initiate disconnect. When a link loss is detected or when an alert is written on the control point, the user interface exposes the alert level on the RGB LED as stated previously.

Also, the battery level and various device information is displayed for reading. The IoT Toolbox can be used to showcase the profile functionality, as shown in the image below.

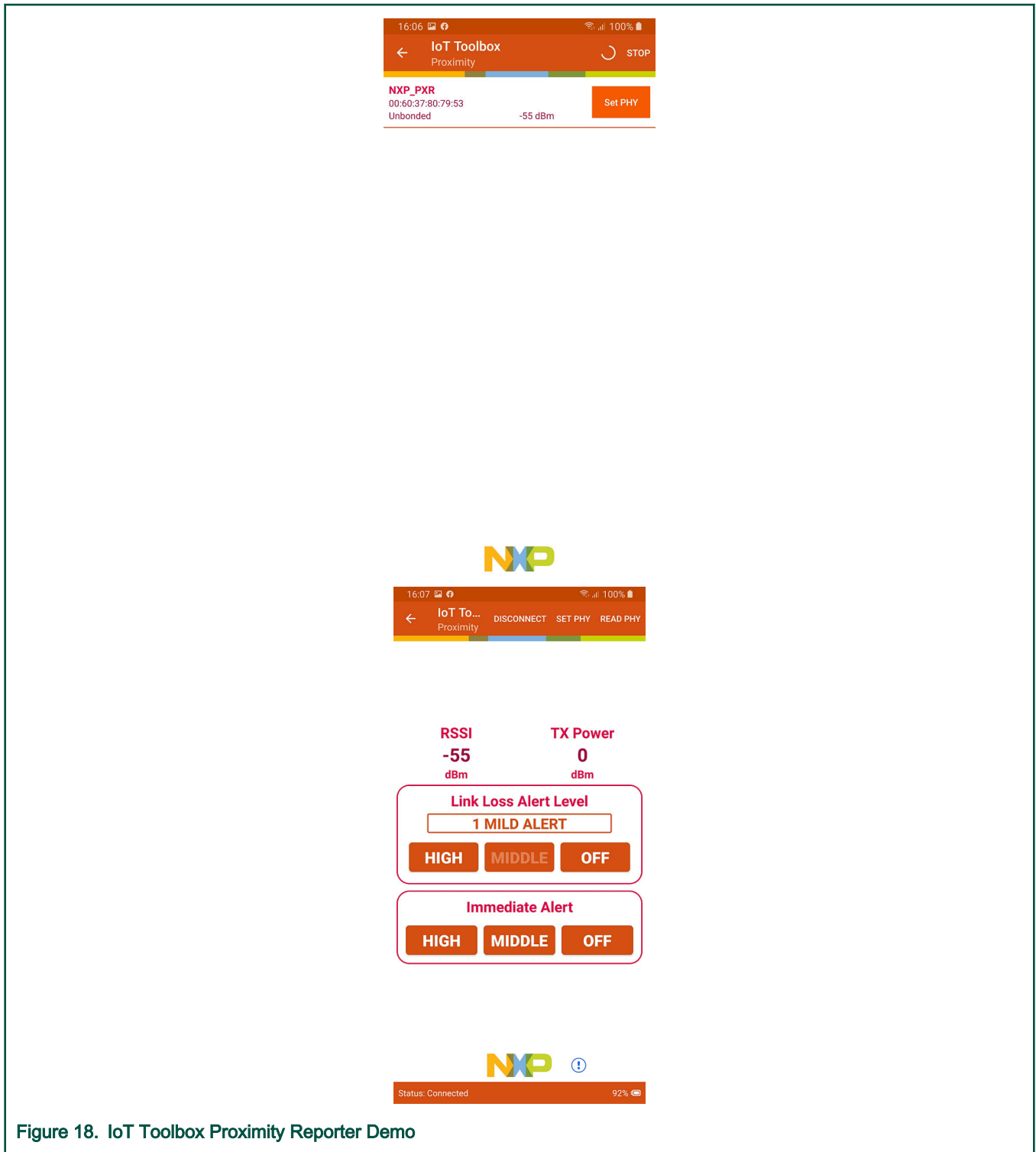


Figure 18. IoT Toolbox Proximity Reporter Demo

## 5.6 HID Device (Mouse)

This section describes implemented profiles and services, user interactions, and testing methods for the HID mouse application.

### 5.6.1 Implemented profiles and services

The HID Device application implements a GATT server and the following profile and services.

- HID over GATT Profile v1.0
- Human Interface Device Service v1.0
- Battery Service v1.0
- Device Information Service v1.1

The application behaves as a GAP peripheral node. It enters GAP General Discoverable Mode and waits for a GAP central node to connect. Security on the services and bonding is enabled on this device.

When the GATT client configures notification, the application starts sending HID reports every two seconds with the movement of the MOUSE\_STEP. The demo moves the cursor in a square pattern between AXIS\_MIN and AXIS\_MAX. The report contains 3 bytes, one for button status, one for X axis, and one for Y axis. The report descriptor matches the example in chapter E.10 from the USB Device Class Definition for Human Interface Devices (USB HID Specification), Version 1.11.

### 5.6.2 Supported platforms

The HID Device application is supported by the following platforms:

- FRDM-KW41Z
- USB-KW41Z
- QN9080CDK
- FRDM-KW36
- FRDM-KW38

### 5.6.3 User interface

After flashing the board, the device is in idle mode (all LEDs flashing). To start advertising, press the ADVSW button. When in GAP Discoverable Mode, CONNLED is flashing. When the central node connects to the peripheral, CONNLED turns solid. To disconnect the node, hold the ADVSW pressed for 2-3 seconds. The node then re-enters GAP Discoverable Mode.

See details below for hardware references.

**Table 6. Hardware references**

Platform	ADVSW	CONNLED
FRDM-KW41Z	SW4	LED3
QN9080CDK	BUTTON1	D5-RED
FRDM-KW36	SW2	LED3
FRDM-KW38	SW2	LED3

### 5.6.4 Usage

The HID mouse can be connected to any Bluetooth Smart Ready products available on the market that supports HID devices or to another supported platform running the HID Host example (setup steps detailed in the HID Host section).

To make the HID mouse visible, press the ADVSW button to start sending advertisements, which causes CONNLED to start flashing. The sensor name "NXP\_HID" shows on the device when its scanning is active. A solid CONNLED indicates a successful connection between the 2 devices. When prompted to enter the pin, type the 999999 passkey.



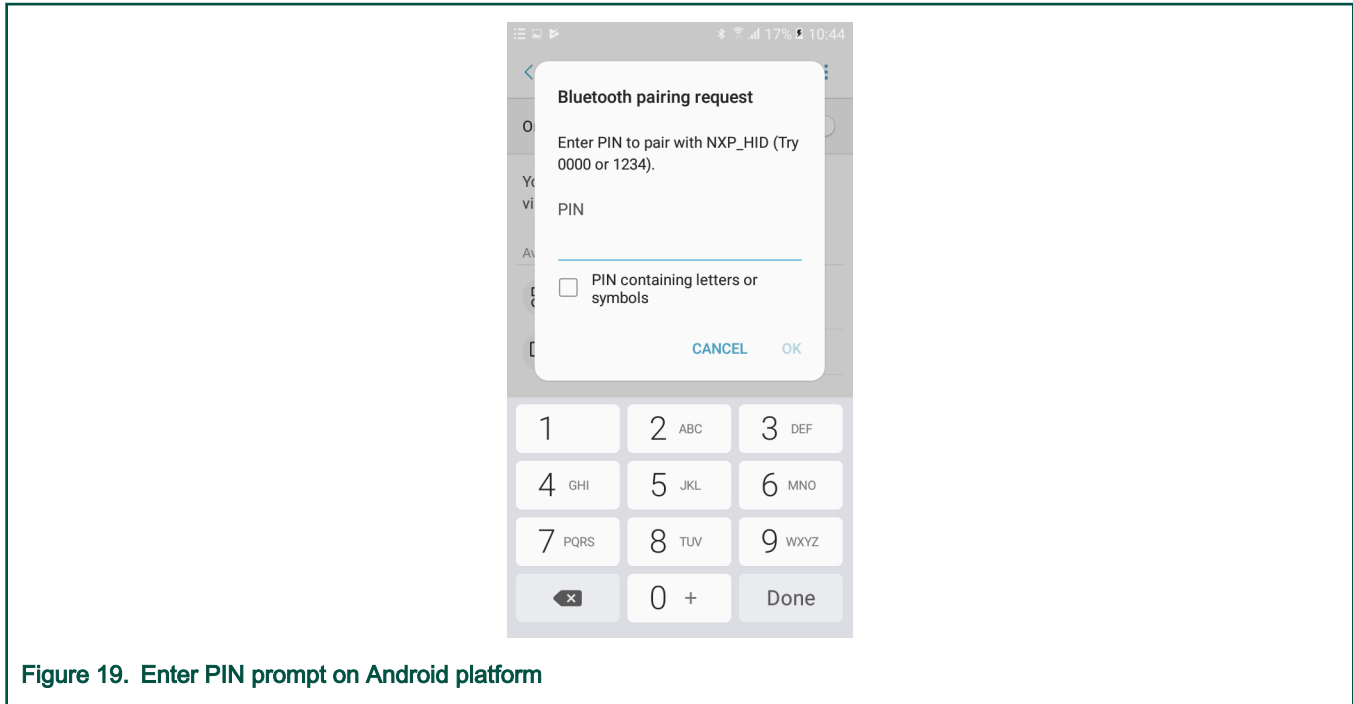


Figure 19. Enter PIN prompt on Android platform

When configured, the HID mouse starts sending HID report, which is configured as explained above, with notifications every 100 milliseconds. The mouse cursor shows a square pattern movement on the screen.

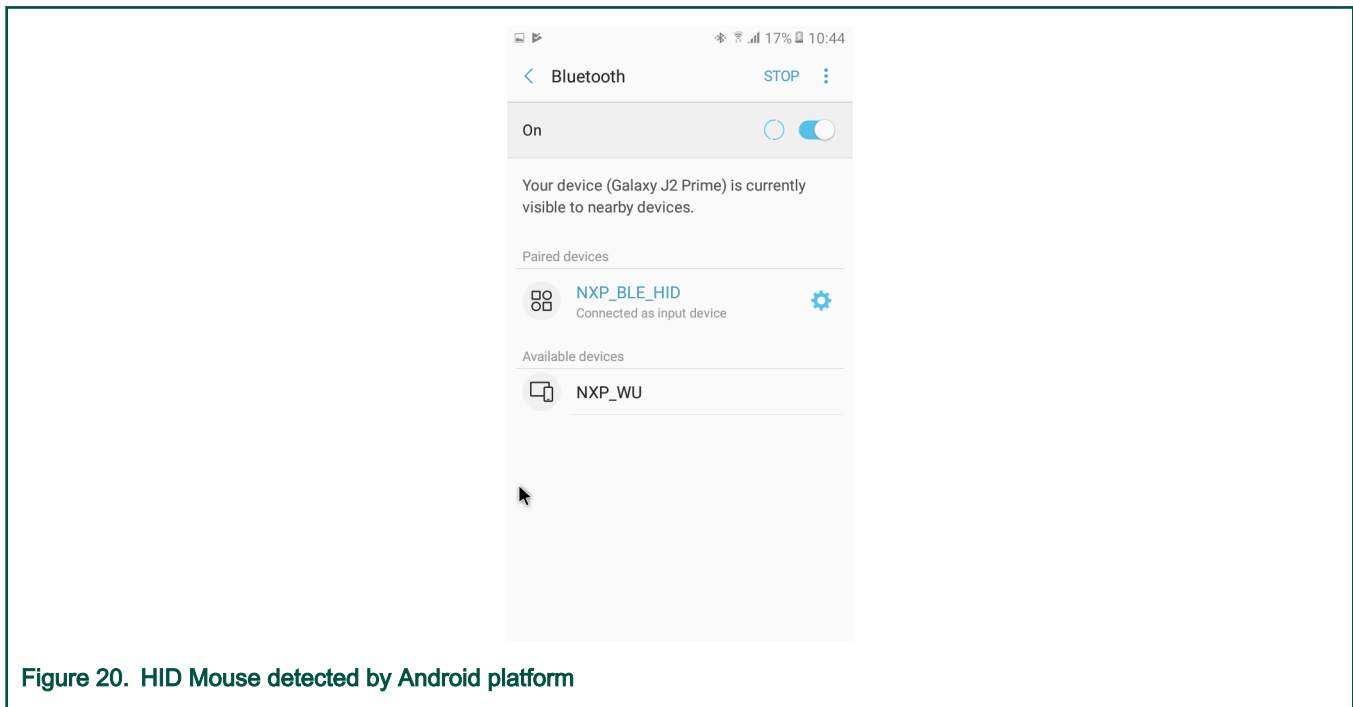


Figure 20. HID Mouse detected by Android platform

## 5.7 HID Host

This section presents the implemented profiles and services, user interactions, and testing methods for the HID Host application.

### 5.7.1 Implemented profiles and services

The HID Host application implements a GATT client or server for the following profile and service.

- HID over GATT Profile v1.0
- Battery Service v1.0
- Device Information Service v1.1

The application behaves as a GAP central node. It enters the GAP Limited Discovery Procedure and searches for HID devices to connect to. After connecting with the peripheral, it configures notifications and displays the received HID reports on a terminal connected to the UART port. The application uses pairing with bonding by default. When connected with the HID Device application, it sends the 999999 passcode to the host stack by default.

### 5.7.2 Supported platforms

The HID Host application is supported by the following platforms:

- FRDM-KW41Z
- USB-KW41Z
- QN9080CDK
- FRDM-KW36
- FRDM-KW38
- USB-KW38

### 5.7.3 User interface

After flashing the board, the device is in idle mode (all LEDs flashing). To start scanning, press the SCANSW button. When in GAP Limited Discovery Procedure, CONNLED is flashing. When the central node connects to the peripheral, CONNLED turns solid. To disconnect the node, hold the SCANSW button pressed for 2-3 seconds. The node then re-enters GAP Limited Discovery Procedure.

See details below for hardware references.

**Table 7. Hardware references**

Platform	SCANSW	CONNLED
FRDM-KW41Z	SW4	LED3
QN9080CDK	BUTTON1	D5-RED
FRDM-KW36	SW2	LED3
FRDM-KW38	SW2	LED3
USB-KW38	SW1	LED2

### 5.7.4 Usage

The application is built to work only with the HID Device application presented in the previous subchapter and it supports up to 2 peripherals connected at the same time.

1. Open a serial port terminal and connect it to board, in the same manner described in section 5.1.3. The start screen is displayed after the board is reset.
2. Press the SCANSW button on the HID Host board to start scanning for devices. Do the same on the HID device board to make it enter discoverable mode. The host connects with the board after it sees it advertise the HID service, connects to it, and configures report notifications. The device then starts sending HID reports, as shown below.

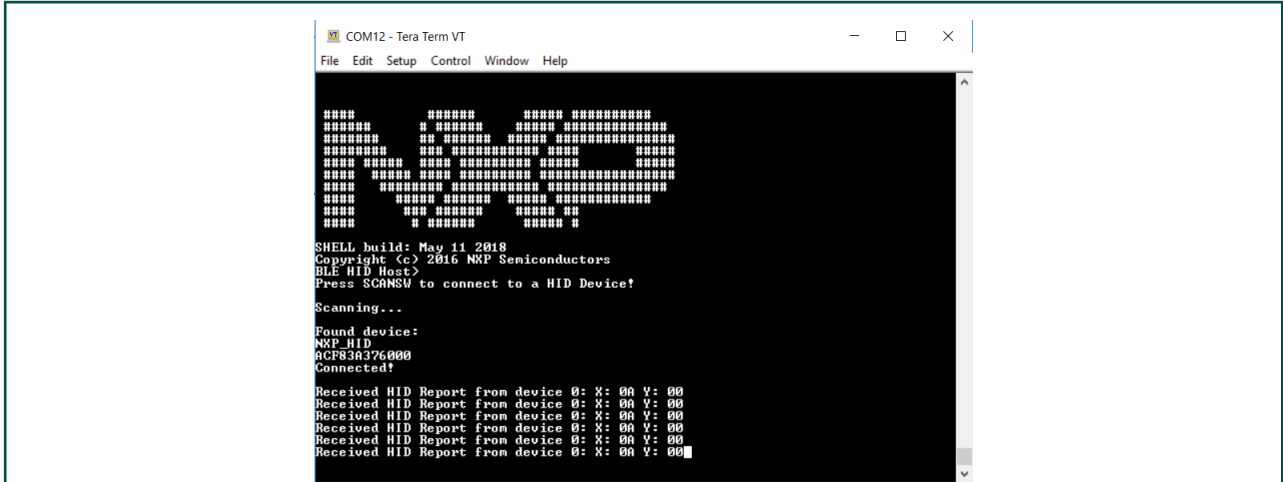


Figure 21. Tera Term – Output Console on HID Host with 1 peripheral connected

- To connect a second HID device, press again the SCANSW button on the HID Host board to start scanning for devices. Do the same on the second HID device board to make it enter discoverable mode. The host connects with the board after it sees it advertise the HID service, connects to it, and configures report notifications. The device then starts sending HID reports. The console will print reports from both devices, as shown below.

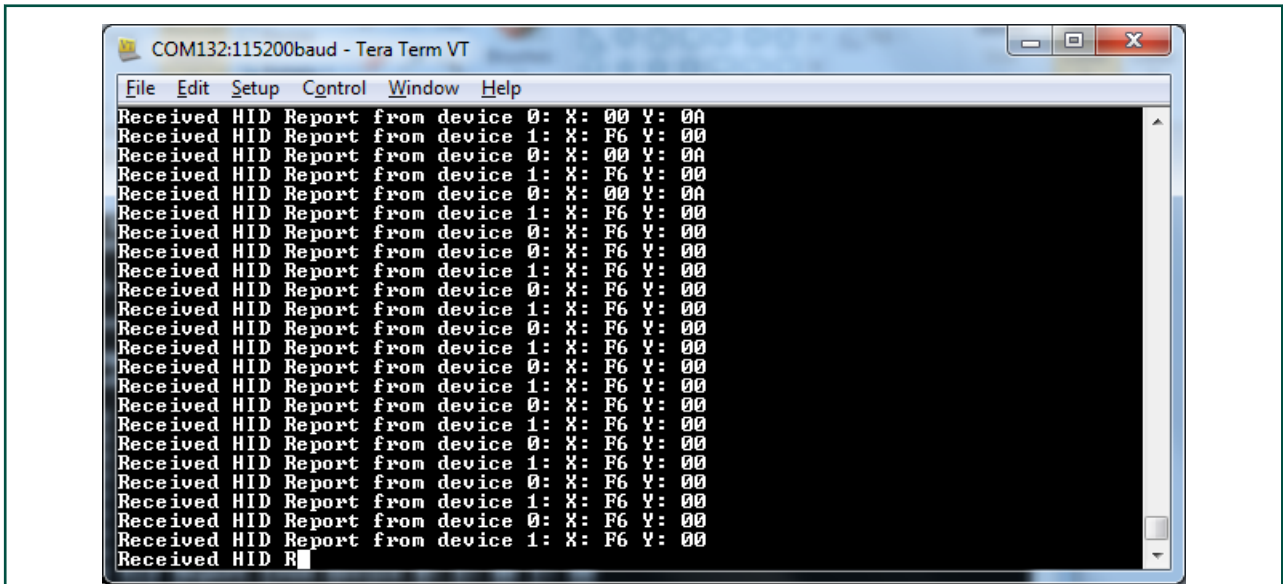


Figure 22. Tera Term – Output Console on HID Host with 2 peripherals connected

## 5.8 Private Profile Server

This section describes the implemented profiles and services, user interactions, and testing methods for the Private Profile Server application.

### 5.8.1 Implemented profiles and services

The Private Profile Server application implements a GATT client or server for the following profile and service.

- Private Profile (UUID: 0000fee9-0000-1000-8000-00805f9b34fb)
- Battery Service v1.0

- Device Information Service v1.1

The application behaves as a GAP peripheral node. It enters GAP General Discoverable Mode and waits for a GAP central node to connect.

## 5.8.2 Supported platforms

The Private Profile Server application is supported by the following platforms:

- QN9080CDK

## 5.8.3 User interface

After flashing the board, the device is in idle mode (all LEDs flashing). To start advertising, press the ADVSW button. When in GAP Discoverable Mode, CONNLED is flashing. When the central node connects to the peripheral, CONNLED turns solid. To disconnect the node, hold the ADVSW pressed for 2-3 seconds. The node then re-enters GAP Discoverable Mode.

See details below for hardware references.

**Table 8. Hardware references**

Platform	ADVSW	CONNLED
QN9080CDK	BUTTON1	D5-RED

## 5.8.4 Usage

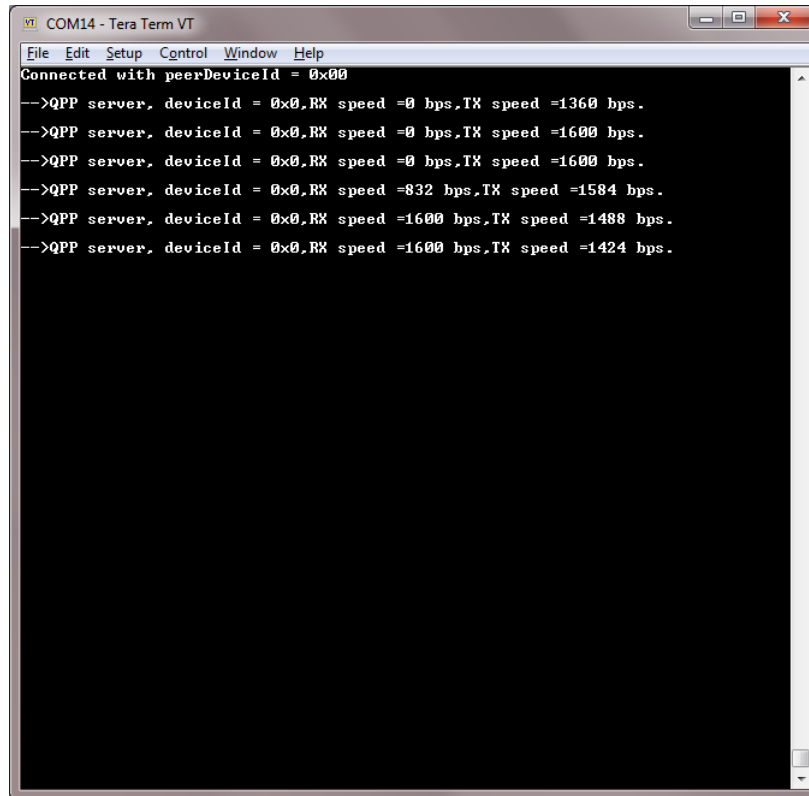
The Private Profile Server can be connected to any supported platforms running Private Profile Client application.

To make the Private Profile Server visible, press the ADVSW button to start sending advertisements, which causes CONNLED to start flashing. The device is visible with the name "NXP\_QPP". A solid CONNLED indicates a successful connection between 2 devices.

The Private Profile is a custom service that implements two custom characteristic:

- Rx Characteristic (UUID: d44bc439-abfd-45a2-b575-925416129600), used to write data to the server
- Tx Characteristic (UUID: d44bc439-abfd-45a2-b575-925416129601), used to get data from the server through notifications

When in connection and if notifications are configured, the Private Profile Server starts sending data every 100 milliseconds through GATT instant value notifications. If the client starts writing the Rx characteristic, the value length received will be monitored in the data Rx callback. Every 10 seconds, a throughput statistics is made and Tx/Rx results are printed on the serial interface.

A screenshot of a Tera Term VT terminal window titled "COM14 - Tera Term VT". The window has a menu bar with "File", "Edit", "Setup", "Control", "Window", and "Help". The terminal output shows a connection to a "QPP server" with a peerDeviceId of 0x00. The output consists of seven lines of log messages, each starting with "-->QPP server, deviceId = 0x0, RX speed = 0 bps, TX speed = [value] bps.". The TX speed values are 1360, 1600, 1600, 1584, 1488, and 1424 bps. The window has standard Windows-style window controls (minimize, maximize, close) in the top right corner.

```
COM14 - Tera Term VT
File Edit Setup Control Window Help
Connected with peerDeviceId = 0x00
-->QPP server, deviceId = 0x0,RX speed =0 bps,TX speed =1360 bps.
-->QPP server, deviceId = 0x0,RX speed =0 bps,TX speed =1600 bps.
-->QPP server, deviceId = 0x0,RX speed =0 bps,TX speed =1600 bps.
-->QPP server, deviceId = 0x0,RX speed =832 bps,TX speed =1584 bps.
-->QPP server, deviceId = 0x0,RX speed =1600 bps,TX speed =1488 bps.
-->QPP server, deviceId = 0x0,RX speed =1600 bps,TX speed =1424 bps.
```

Figure 23. Private Profile Server – Output Console

## 5.9 Private Profile Client

This section describes the implemented profiles and services, user interactions, and testing methods for the Private Profile Client application.

### 5.9.1 Implemented profiles and services

The Private Profile Client application implements a GATT client or server for the following profile and service.

- Private Profile (UUID: 0000fee9-0000-1000-8000-00805f9b34fb)
- Battery Service v1.0
- Device Information Service v1.1

The application behaves as a GAP central node. It enters the GAP Limited Discovery Procedure and searches for Private Profile servers to connect to. After connecting with the peripheral, it configures notifications and monitors the data received through notifications.

### 5.9.2 Supported platforms

The Private Profile Client application is supported by the following platforms:

- QN9080CDK

### 5.9.3 User interface

After flashing the board, the device is in idle mode (all LEDs flashing). To start scanning, press the SCANSW button. When in GAP Limited Discovery Procedure, CONNLED is flashing. When the central node connects to the peripheral, CONNLED turns

solid. To disconnect the node, hold the SCANSW button pressed for 2-3 seconds. The node then re-enters GAP Limited Discovery Procedure.

See details below for hardware references.

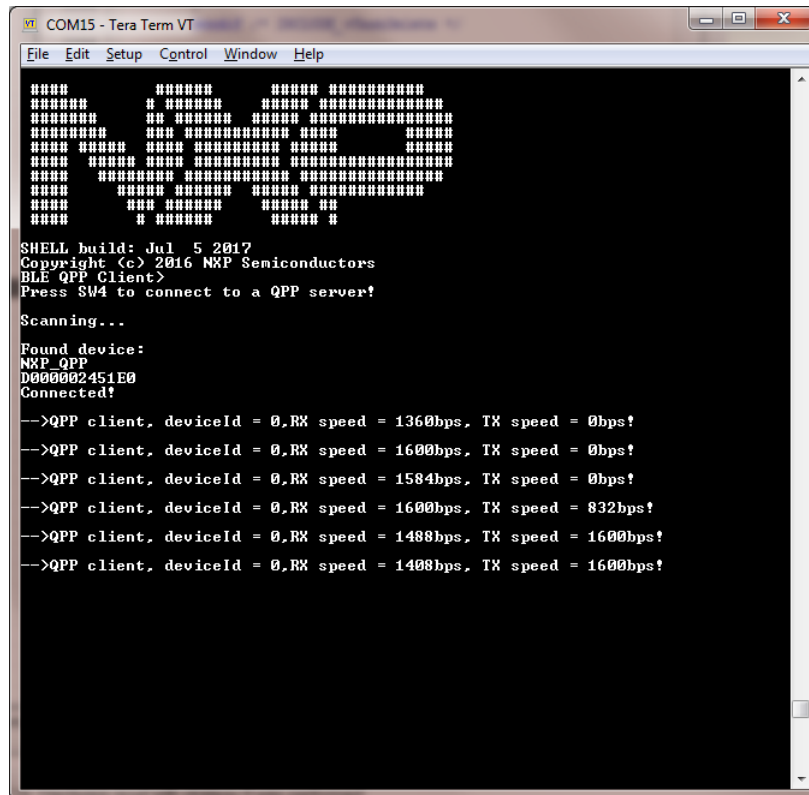
**Table 9. Hardware references**

Platform	SCANSW	CONNLED	SENDSW
QN9080CDK	BUTTON1	D5-RED	BUTTON2

### 5.9.4 Usage

The application is built to work only with the Private Profile Server application presented in the previous subchapter.

1. Open a serial port terminal and connect it to board, in the same manner described in section 5.1.3. The start screen is displayed after the board is reset.
2. Press the SCANSW button on the Private Profile Client board to start scanning for devices. Press the ADVSW button on the Private Profile Server board to make it enter discoverable mode. The client connects to the server after it sees it advertise the Private Profile Service. The client then configures notifications and monitors data received through notifications.
3. If the user wants to also send data from the client to the server, after pressing SENDSW, the client starts writing the Rx characteristic every 100 milliseconds.



**Figure 24. Private Profile Client – Output Console**

## 5.10 Cycling Speed and Cadence Sensor

This section describes the implemented profiles and services, user interactions, and testing methods for the Cycling Speed and Cadence Sensor application.

### 5.10.1 Implemented profiles and services

The Cycling Speed and Cadence application implements a GATT server and the following profile and services.

- Cycling Speed and Cadence Profile v1.0
- Cycling Speed and Cadence Service v1.0
- Battery Service v1.0
- Device Information Service v1.1

The application behaves as a GAP peripheral node. It enters GAP General Discoverable Mode and waits for a GAP central node to connect. When the collector configures notifications, the sensor starts sending speed and cadence measurements. The measurement values are generated randomly on the device.

### 5.10.2 Supported platforms

The Cycling Speed and Cadence Sensor application is supported by the following platforms:

- FRDM-KW41Z
- USB-KW41Z
- QN9080CDK
- FRDM-KW36
- FRDM-KW38

### 5.10.3 User interface

After flashing the board, the sensor is in idle mode (all LEDs flashing). To start advertising, press the ADVSW button. When in GAP Discoverable Mode, CONNLED is flashing. When the central node connects to the peripheral, CONNLED turns solid. To disconnect the node, hold the ADVSW button pressed for 2-3 seconds. The node then re-enters GAP Discoverable Mode

See details below for hardware references.

**Table 10. Hardware references**

Platform	ADVSW	CONNLED
FRDM-KW41Z	SW4	LED3
QN9080CDK	BUTTON1	D5-RED
FRDM-KW36	SW2	LED3
FRDM-KW38	SW2	LED3

### 5.10.4 Usage

The sensor can be connected to any Bluetooth Smart Ready products available on the market. To make the sensor visible, press the ADVSW button to start sending advertisements, which causes CONNLED to start flashing. The sensor name "NXP\_CSCS" shows on the device when its scanning is active. A solid CONNLED indicates a successful connection between the 2 devices. Hold the ADVSW button pressed for 2-3 seconds anytime to initiate disconnect.

If configured, the sensor notifies the application with speed and cadence measurements every second. Also, the battery level and various device information is exposed for reading. The IoT Toolbox can be used to showcase the profile functionality, as shown in the image below.

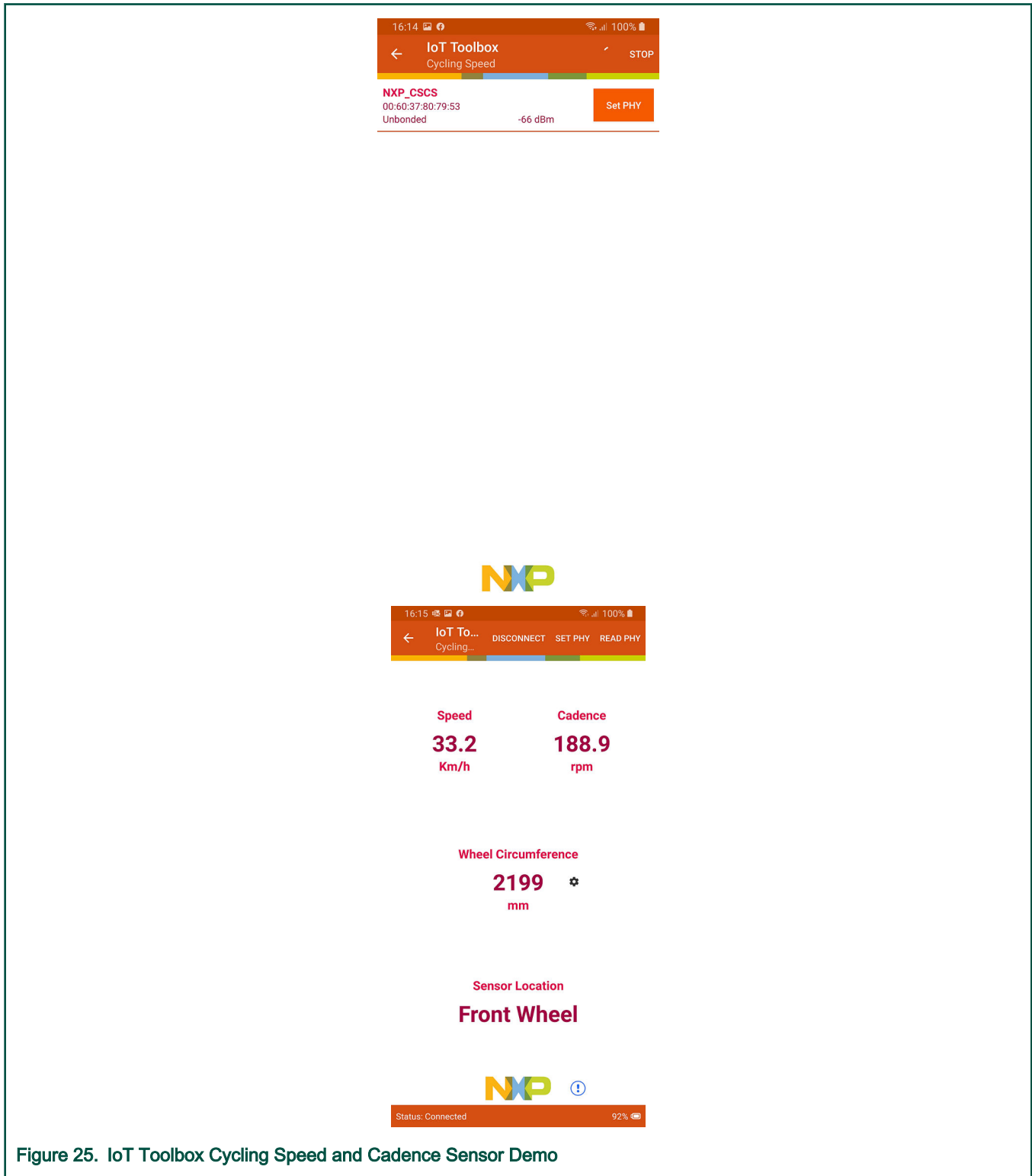


Figure 25. IoT Toolbox Cycling Speed and Cadence Sensor Demo



## 5.11 Cycling Power Sensor

This section presents the implemented profiles and services, user interactions, and testing methods for the Cycling Power Sensor application.

### 5.11.1 Implemented profiles and services

The Cycling Power Sensor application implements a GATT server and the following profile and services.

- Cycling Power Profile v1.0
- Cycling Power Service v1.0
- Battery Service v1.0
- Device Information Service v1.1
- Current Time Service v1.1

The application behaves as a GAP peripheral node. It enters GAP General Discoverable Mode and waits for a GAP central node to connect. When the collector configures notifications and/or indications, the sensor starts sending power measurements. The measurement values are generated randomly on the device.

### 5.11.2 Supported platforms

The Cycling Power Sensor application is supported by the following platforms:

- FRDM-KW41Z
- USB-KW41Z
- QN9080CDK
- FRDM-KW36
- FRDM-KW38

### 5.11.3 User interface

After flashing the board, the sensor is in idle mode (all LEDs flashing). To start advertising, press the ADVSW button. When in GAP Discoverable Mode, CONNLED is flashing. When the central node connects to the peripheral, CONNLED turns solid. To disconnect the node, hold the ADVSW button pressed for 2-3 seconds. The node then re-enters GAP Discoverable Mode.

Because all the fields in the power measurement cannot fit into one single ATT frame, the user can toggle the optional fields by pressing the TOGSW button.

See details below for hardware references.

**Table 11. Hardware references**

Platform	ADVSW	CONNLED	TOGSW
FRDM-KW41Z	SW4	LED3	SW3
QN9080CDK	BUTTON1	D5-RED	BUTTON2
FRDM-KW36	SW2	LED3	SW3
FRDM-KW38	SW2	LED3	SW3

### 5.11.4 Usage

The sensor can be connected to any Bluetooth Smart Ready products available on the market. To make the sensor visible, press the ADVSW button to start sending advertisements, which causes CONNLED to start flashing. The sensor name "NXP\_CPS"

shows on the device when its scanning is active. A solid CONNLED indicates a successful connection between the 2 devices. Hold the ADVSW button pressed for 2-3 seconds anytime to initiate disconnect. If configured, the sensor notifies the application with power measurements. Also, the battery level and various device information is displayed for reading.

To demonstrate profile functionality, any bike application (from Health & Fitness category) can be installed on Apple iOS or Android OS handheld devices that support Bluetooth Low Energy.

## 5.12 Running Speed and Cadence Sensor

This section describes the implemented profiles and services, user interactions, and testing methods for the Running Speed and Cadence Sensor application.

### 5.12.1 Implemented profiles and services

The Running Speed and Cadence application implements a GATT server and the following profile and services.

- Running Speed and Cadence Profile v1.0
- Running Speed and Cadence Service v1.0
- Battery Service v1.0
- Device Information Service v1.1

The application behaves as a GAP peripheral node. It enters GAP General Discoverable Mode and waits for a GAP central node to connect. When the collector configures notifications, the sensor starts sending speed and cadence measurements. The measurement values are generated randomly on the device.

### 5.12.2 Supported platforms

The Running Speed and Cadence Sensor application is supported by the following platforms:

- FRDM-KW41Z
- USB-KW41Z
- QN9080CDK
- FRDM-KW36
- FRDM-KW38

### 5.12.3 User interface

After flashing the board, the sensor is in idle mode (all LEDs flashing). To start advertising, press the ADVSW button. When in GAP Discoverable Mode, CONNLED is flashing. When the central node connects to the peripheral, CONNLED turns solid. To disconnect the node, hold the ADVSW button pressed for 2-3 seconds. The node then re-enters GAP Discoverable Mode.

When the node is connected, the user can toggle the running status (running/walking) by pressing the TOGSW button. A LED is turned on solid to denote the 'walking' status, this LED is turned off in 'running' status. On platforms other than QN9090/K32W061 DK6, the RGB LED is used for that purpose, whereas on QN9090/K32W061 DK6, the DS3 red LED is used.

See details below for hardware references.

**Table 12. Hardware references**

Platform	ADVSW	CONNLED	TOGSW
FRDM-KW41Z	SW4	LED3	SW3
QN9080CDK	BUTTON1	D5-RED	BUTTON2

*Table continues on the next page...*

**Table 12. Hardware references (continued)**

Platform	ADVSW	CONNLED	TOGSW
FRDM-KW36	SW2	LED3	SW3
FRDM-KW38	SW2	LED3	SW3

### 5.12.4 Usage

The sensor can be connected to any Bluetooth® Smart Ready products available on the market. To make the sensor visible, press the ADVSW button to start sending advertisements, which causes CONNLED to start flashing. The sensor name “NXP\_RSCS” shows on the device when its scanning is active. A solid CONNLED indicates a successful connection between the two devices. Hold the ADVSW button pressed for 2-3 seconds anytime to initiate disconnect.

If configured, the sensor notifies the application with speed and cadence measurements every second. Also, the battery level and various device information are exposed for reading. The IoT Toolbox can be used to showcase the profile functionality, as shown in the image below.

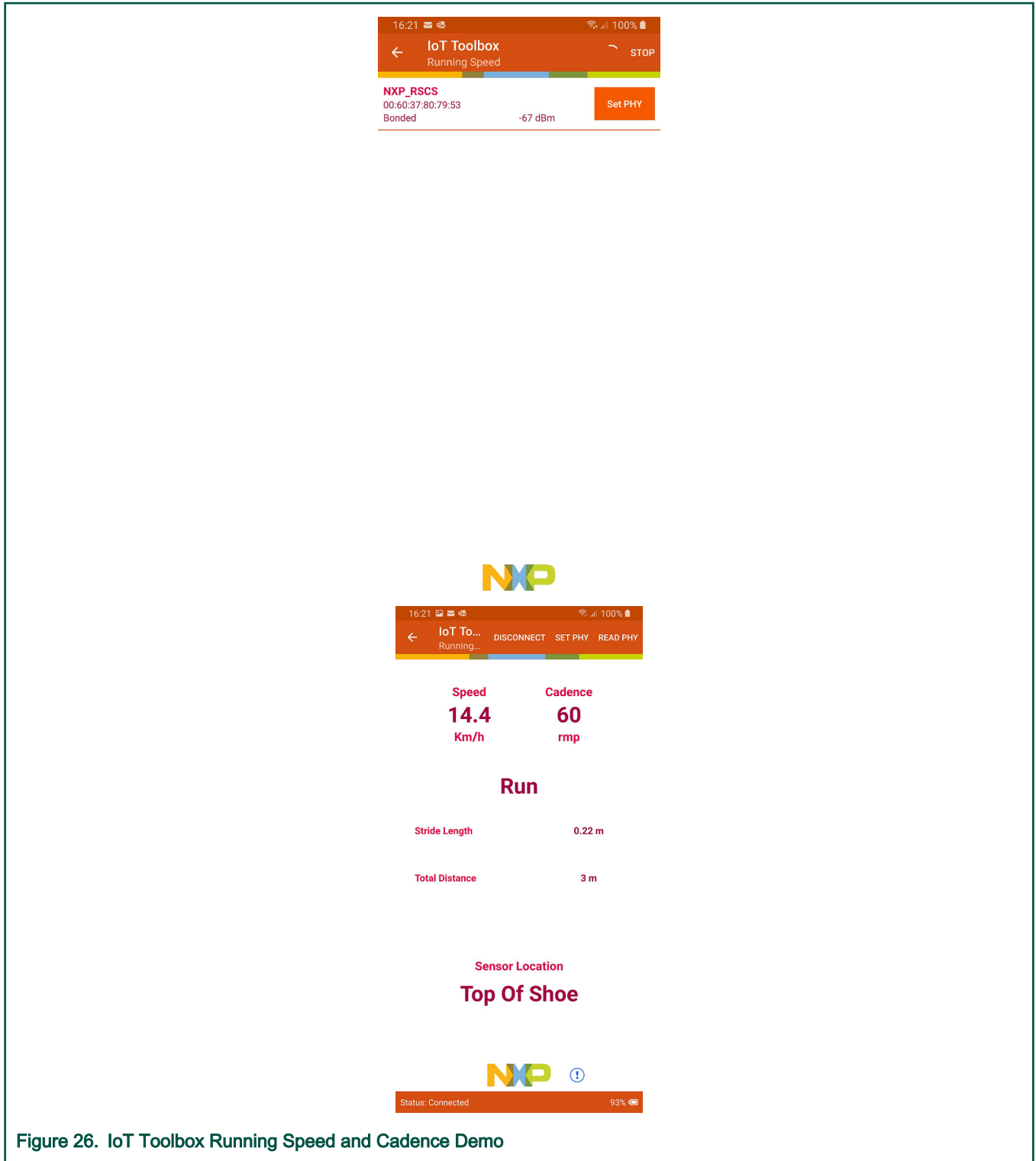


Figure 26. IoT Toolbox Running Speed and Cadence Demo

### 5.13 Health Thermometer

This section presents the implemented profiles and services, user interactions, and testing methods for the Health Thermometer application.

### 5.13.1 Implemented profiles and services

The Health Thermometer application implements a GATT server and the following profile and services.

- Health Thermometer Profile v1.0
- Health Thermometer Service v1.0
- Battery Service v1.0
- Device Information Service v1.1
- Current Time Service v1.1

The application behaves as a GAP peripheral node. It enters GAP General Discoverable Mode and waits for a GAP central node to connect. When the collector configures notifications or indications, the sensor starts sending intermediate temperature and temperature measurements.

In a 5 second interval, the sensor sends 5 events (4 notifications with the intermediate temperature and one indication with the temperature measurement). The intermediate temperature and temperature measurements are generated randomly on the device.

### 5.13.2 Supported platforms

The Health Thermometer application is supported by the following platforms:

- FRDM-KW41Z
- USB-KW41Z
- QN9080CDK
- FRDM-KW36
- FRDM-KW38

### 5.13.3 User interface

After flashing the board, the sensor is in idle mode (all LEDs flashing). To start advertising, press ADVSW. When in GAP Discoverable Mode CONNLED is flashing. When the central node connects to the peripheral CONNLED turns solid. Hold the ADVSW button pressed for 2-3 seconds anytime to initiate disconnect. The node re-enters the GAP Discoverable Mode.

See details below for hardware references.

**Table 13. Hardware references**

Platform	ADVSW	CONNLED
FRDM-KW41Z	SW4	LED3
QN9080CDK	BUTTON1	D5-RED
FRDM-KW36	SW2	LED3
FRDM-KW38	SW2	LED3

### 5.13.4 Usage

The sensor can be connected to any Bluetooth® Smart Ready products available on the market. To make the sensor visible, press the ADVSW button to start sending advertisements, which causes CONNLED to start flashing. The sensor name “NXP\_HTS” shows on the device when its scanning is active. A solid CONNLED indicates a successful connection between the two devices. Hold the ADVSW button pressed for 2-3 seconds anytime to initiate disconnect.

If configured, the sensor notifies the application with temperature measurements. Also, the battery level and various device information are exposed for reading. The IoT Toolbox can be used to showcase the profile functionality, as shown in the image below.



Figure 27. IoT Toolbox Health Thermometer Demo

## 5.14 Low Power Temperature Sensor and Collector

This section describes the implemented profiles and services, user interactions, and testing methods for the temperature sensor application.

### 5.14.1 Implemented profiles and services

The Temperature Sensor application implements a GATT server, a custom profile and the following services.

- Temperature Service (UUID: 01ff0200-ba5e-f4ee-5ca1-eb1e5e4b1ce0)
- Battery Service v1.0
- Device Information Service v1.1

The application behaves as a GAP peripheral node. It enters GAP General Discoverable Mode and waits for a GAP central node to connect and configure notifications for the temperature value.

The Temperature service is a custom service that implements the Temperature characteristic (UUID: 0x2A6E) with a Characteristic Presentation Format descriptor (UUID: 0x2904), both defined by the Bluetooth SIG.

The Temperature Collector application implements a GATT client or server for the following profile and services.

- Temperature Service (UUID: 01ff0200-ba5e-f4ee-5ca1-eb1e5e4b1ce0)
- Battery Service v1.0
- Device Information Service v1.1

The application behaves as a GAP central node. It enters GAP Limited Discovery Procedure and searches for sensor devices to pair with. After pairing with the peripheral, it configures notifications and displays temperature values on a terminal connected to the UART port.

Both application uses pairing with bonding by default. When connected with the Low Power Temperature Sensor application, the collector sends the 999999 passcode to the host stack by default.

### 5.14.2 Supported platforms

The Temperature Sensor and Collector applications are supported by the following platforms:

- FRDM-KW41Z
- USB-KW41Z
- QN9080CDK
- FRDM-KW38
- FRDM-KW36

#### NOTE

On FRDM-KW36 and FRDM-KW38, deep-sleep modes 5 and 8 are used by default.

### 5.14.3 User interface

After flashing the board, both nodes enter cPWR\_DeepSleepMode deep-sleep mode (3- LLS or 5 - VLLS). To flash the board in case the sensor is put in deep sleep, press WAKESW or RESET. The application default configuration enables low power which disables LED support. The user can manually change this and enable LED support, else all subsequent LED behavior references are ignored. All LEDs are off. When the node is awake and communicating, CONNLED is on. To wake up the node press the WAKESW button. For FRDM-KW38 board, the default enabled low-power modes are 1 and 5. This means that the MCU is put into VLLS2 or VLLS3 stop mode. In this mode, the LED module operation is not implemented.

See details below for hardware references.

**Table 14. Hardware references**

Platform	WAKESW	CONNLED
FRDM-KW41Z	SW4	LED3
QN9080CDK	BUTTON1	D5-RED
FRDM-KW36	SW3	LED3
FRDM-KW38	SW3	LED3

### 5.14.4 Usage

The setup requires two supported platforms, one for the temperature sensor and one for the temperature collector.

1. Open a serial port terminal and connect it to the temperature collector board, in the same manner as described in 5.1.3. The start screen is displayed after the board is reset. At first the LEDs are off on both devices.
2. To start advertising on the sensor, press the WAKESW button and CONNLED lights up. The sensor enters into the `gAppDeepSleepMode_c` deep-sleep mode (1 - LLS or 8 - VLLS), which means that the MCU wakes up on any packet from the Link layer, in this case the connect request. If no connection is established in an interval of 30 seconds, the sensor stops advertising and enters into the `cPWR_DeepSleepMode` again. CONNLED turns off.
3. To start scanning on the collector, press the WAKESW button and CONNLED lights up. The device wakes up, enters into the `cPWR_DeepSleepMode` deep-sleep mode 6, scans and connects to a compatible sensor device. If no connection is established within 30 seconds, the collector stops scanning and enters deep-sleep mode (for QN9090/ K32W061 is deep sleep mode 3) again. CONNLED turns off.
4. If the collector connects to a sensor node, it bonds (if no bond was previously made), does service discovery (only the first time it connects with the sensor) and configures notification and waits for notifications from the sensor for 5 seconds. If no data is sent, the node disconnects and re-enters `cPWR_DeepSleepMode`. The sensor exits low power and sends a notification with the value of the temperature read through an ADC from the thermistor, if present, or random generated if not.

Once the connection is established the PHY is automatically updated to 2 M, if both the sensor and the collector support this feature. The PHY update is configurable from the application.



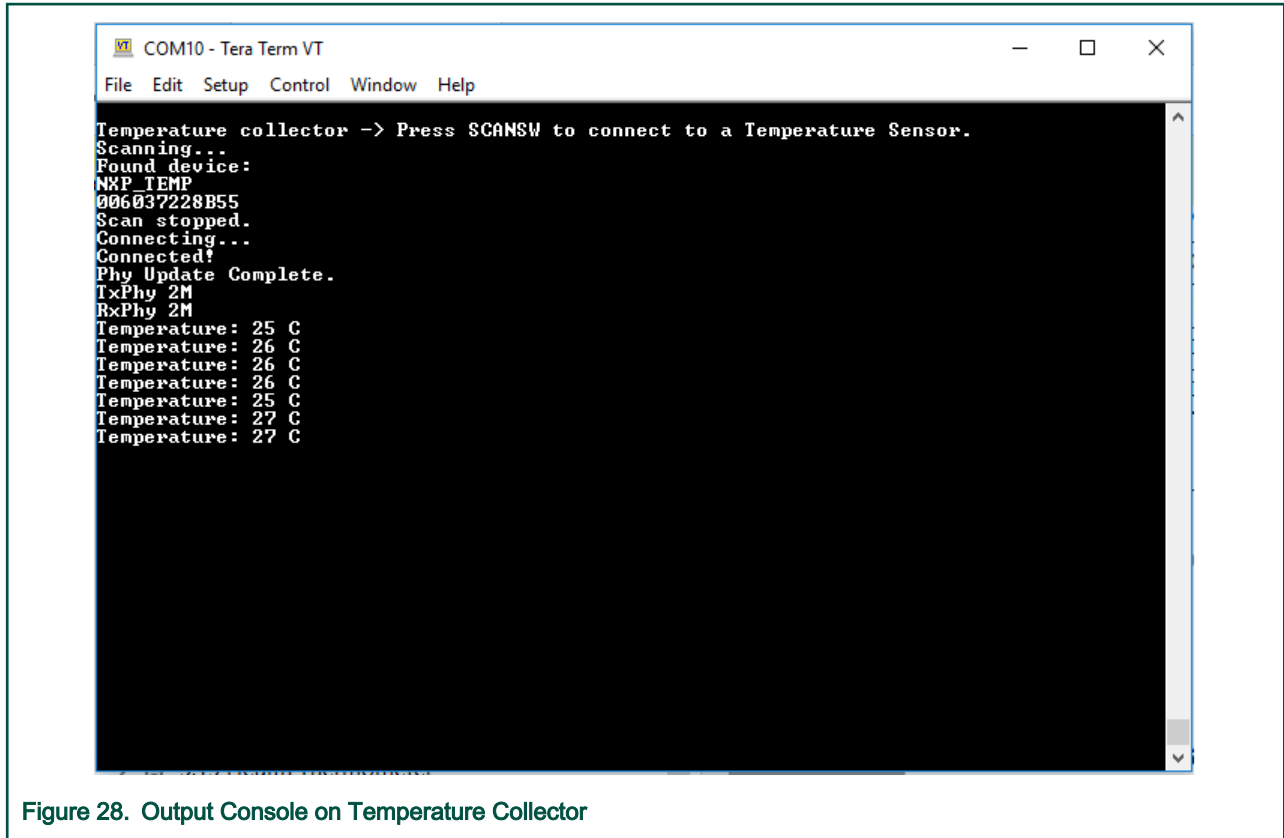


Figure 28. Output Console on Temperature Collector

- Subsequent key pressing triggers other notifications for the collector. If no key is pressed in an interval of 5 seconds, the sensor node disconnects and re-enters cPWR\_DeepSleepMode.

## 5.15 Low-Power Extended Advertising Peripheral and Extended Advertising Central (KW37 platforms)

This section describes the implemented profiles and services, user interactions, and testing methods for the `adv_ext_peripheral` and `adv_ext_central` applications.

### 5.15.1 Implemented profile and services

The `adv_ext_peripheral` application implements a GATT server, a custom profile and the following services.

- Temperature Service (UUID: 01ff0200-ba5e-f4ee-5ca1-eb1e5e4b1ce0)
- Battery Service v1.0
- Device Information Service v1.1

The application behaves as a GAP peripheral node. It enters GAP General Discoverable Mode and waits for a GAP central node to connect and configure notifications for the temperature value.

The Temperature service is a custom service that implements the Temperature characteristic (UUID: 0x2A6E) with a Characteristic Presentation Format descriptor (UUID: 0x2904), both defined by the Bluetooth SIG.

The `adv_ext_central` application implements a GATT client or server for the following profile and services.

- Temperature Service (UUID: 01ff0200-ba5e-f4ee-5ca1-eb1e5e4b1ce0)
- Battery Service v1.0
- Device Information Service v1.1

The application behaves as a GAP central node. It enters GAP Limited Discovery Procedure and searches for peripherals devices to pair with. After pairing with the peripheral, it configures notifications and displays temperature values on a terminal connected to the UART port.

Both applications use pairing with bonding by default. When connected with the Low-Power Extended Advertising Peripheral application, the Extended Advertising Central application sends the 999999 passcode to the host stack by default.

### 5.15.2 Supported platforms

The Extended Advertising Peripheral and Central applications are supported by the following platforms:

- FRDM-KW38

Deep-sleep modes 3 and 1 are used by default.

### 5.15.3 User interface

After flashing the board, both nodes enter cPWR\_DeepSleepMode deep-sleep mode. To reflash the board, press WAKESW. The application default configuration enables low-power that disables LED support. The user disables low-power and enables LED support setting cPWR\_UsePowerDownMode 0. To wake up the node, press the WAKESW button. Both applications provide guidance over the UART. Open a serial port terminal using the following settings: baud rate 115200, data bits 8, parity none, stop bits 1.

See details below for hardware references.

Hardware references:

**Table 15. Extended advertising peripheral**

Platform	WAKESW	OPTSW	ADVLED	CONNLED
FRDM-KW38	SW3	SW2	LED3	LED4

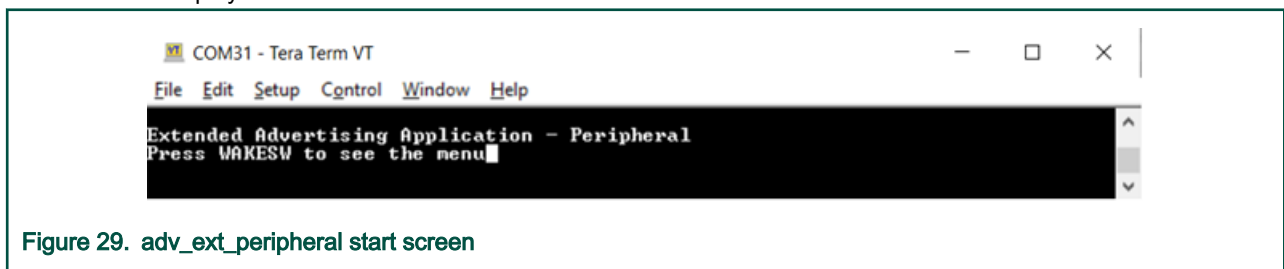
**Table 16. Extended advertising central**

Platform	WAKESW	SCANLED	CONNLED
FRDM-KW38	SW3	LED3	LED4

### 5.15.4 Usage

The setup requires two supported platforms, one for the adv ext peripheral, and one for the adv ext central.

1. Open a serial port terminal and connect it to each platform with the settings provided in the previous paragraph. The start screen is displayed after the boards are reset.



**Figure 29. adv\_ext\_peripheral start screen**

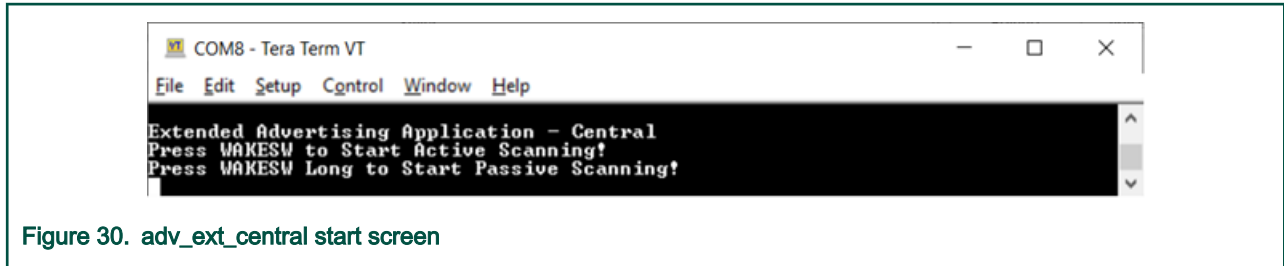


Figure 30. adv\_ext\_central start screen

- On the adv\_ext\_peripheral board, press the WAKESW button. The board exits cPWR\_DeepSleepMode deep sleep mode and prints the menu.

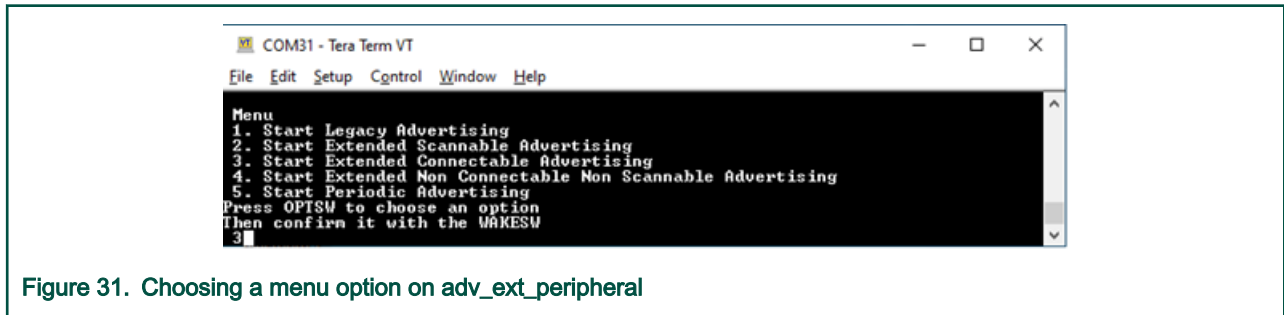


Figure 31. Choosing a menu option on adv\_ext\_peripheral

Use the OPTSW to choose an option. The option printed on the bottom changes every time the switch is pressed. When the option matches your intention (For example, 3 Start Extended Connectable Advertising), press the WAKESW again to make a decision. The advertising type chosen is started, the deep-sleep mode changes to gAppDeepSleepMode\_c and the board starts entering low-power between advertising events. Next time the WAKESW is pressed, an updated menu is printed (For example, at option 3 Stop Extended Connectable Advertising). There is no timeout for advertising. The board continues advertising until it is stopped, or a connection is established (for legacy and extended connectable advertising only) with an adv\_ext\_central device. The connection is terminated five seconds after the central device configures notifications for the temperature value. When all advertisings are off and all connections are terminated, the deep-sleep mode changes to cPWR\_DeepSleepMode and the board enter low-power until the WAKESW is pressed again. When cPWR\_UsePowerDownMode is set 0, LEDs are enabled. The ADVLED flashes whenever an advertising starts and is ON otherwise. The CONNLED flashes whenever there is a connection under way and is ON otherwise.

- On the adv\_ext\_central board, there are only two options: Press WAKESW to start active scanning or long press WAKESW to start passive scanning. If catching extended scannable advertising is not an option, choose passive scanning. Otherwise active scanning must be chosen. The device wakes up, starts scanning, and enters deep-sleep mode 6. The scanning ends when the 60 seconds timeout is reached or when a connection with an adv\_ext\_peripheral device is established. If a connection is established, the central changes its low-power mode to gAppDeepSleepMode\_c. During scanning, all advertisings caught from adv\_ext\_peripheral devices are printed on the terminal window. When an extended nonconnectable, nonscannable advertising with a periodic advertising attached is detected, the adv\_ext\_central device attempts to sync with the periodic advertising train and prints the periodic advertising data on the terminal window. When the 60 seconds timer expires or the connection ends, the device reenters cPWR\_DeepSleepMode until the WAKESW is pressed again and all syncs with periodic advertising trains are terminated. If cPWR\_UsePowerDownMode is set 0, LEDs are enabled. The SCANLED flashes, whenever the device is scanning and is ON otherwise. The CONNLED flashes, whenever there is a connection under way and is ON otherwise.

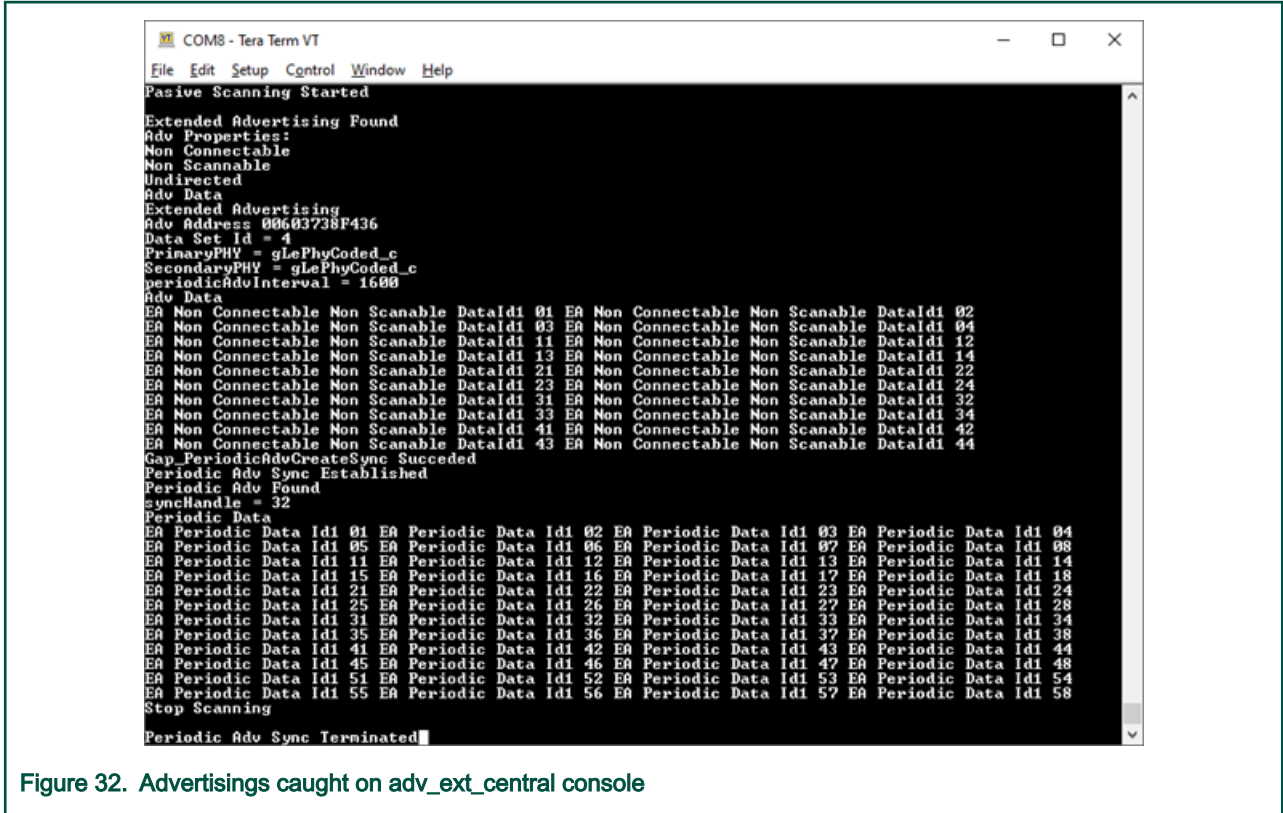


Figure 32. Advertisings caught on adv\_ext\_central console

4. If the adv\_ext\_central connects to an adv\_ext\_peripheral device, it bonds (if no bond was previously made), does service discovery (only the first time it connects with the peripheral), configures notification and waits for notifications from the peripheral. If no data is sent within 5 seconds, the node disconnects and reenters cPWR\_DeepSleepMode. The peripheral sends a notification with the value of the temperature read through an ADC from the thermistor, if present, or random generated if not. When the central receives the notification, it prints it on the terminal window and disconnects in 5 seconds.

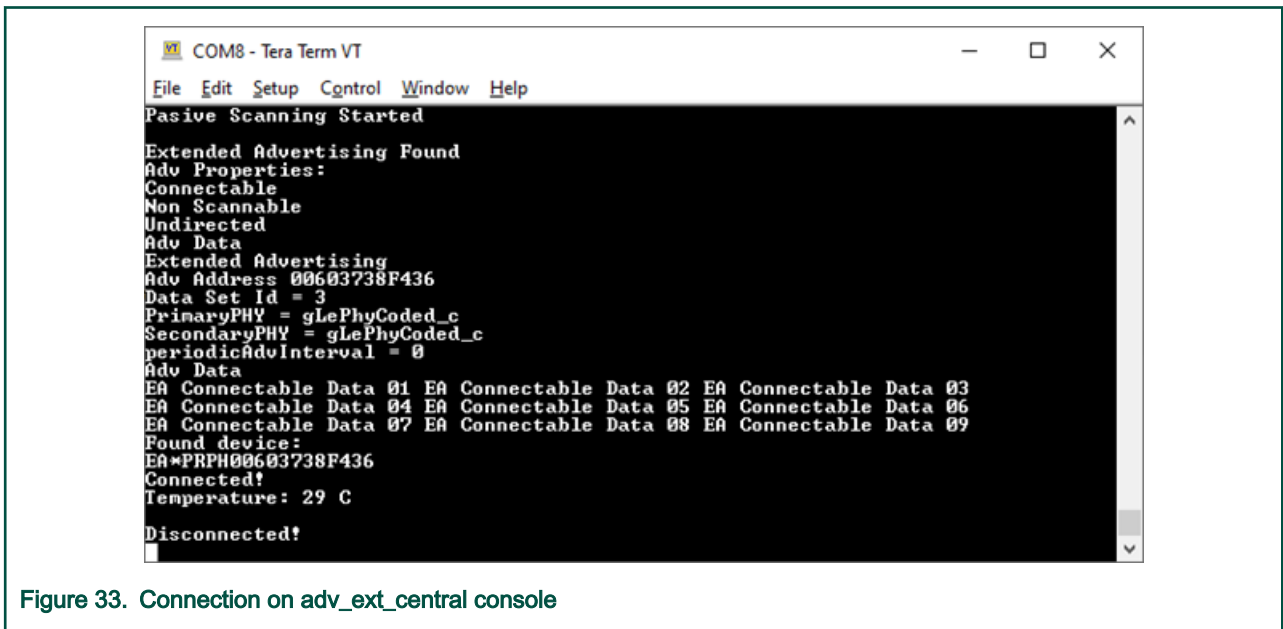


Figure 33. Connection on adv\_ext\_central console

## 5.16 IPv6 Router and Node (KW41 Platforms)

For information about the IPv6 Node and Router applications, see the *Bluetooth® Low Energy Transport for IPv6 Datagrams* (document BLEIP6UG).

## 5.17 Alert Notification Server

This section describes the implemented profiles and services, user interactions, and testing methods for the Alert Notification Server application.

### 5.17.1 Implemented profile and services

The Alert Notification application implements a GATT client or server for the following profile and services.

- Alert Notification Profile v1.0
- Alert Notification Service v1.0
- Battery Service v1.0
- Device Information Service v1.1
- Current Time Service v1.1
- Next DST Change Service v1.0
- Reference Time Update Service v1.0

The application behaves as a GAP central node. It enters the GAP Limited Discovery Procedure and searches for Alert Notification Clients or Time Clients to connect to. After connecting with the peripheral, it waits for notifications to be configured.

### 5.17.2 Supported platforms

The Alert Notification Server application is supported by the following platforms:

- FRDM-KW41Z
- USB-KW41Z
- QN9080CDK
- FRDM-KW36
- FRDM-KW38

### 5.17.3 User interface

After flashing the board, the device is in idle mode (all LEDs flashing). To start scanning, press the SCANSW button. When in GAP Limited Discovery Procedure, CONNLED is flashing. When the central node connects to the peripheral, CONNLED turns solid. To disconnect the node, hold the SCANSW button pressed for 2-3 seconds. The node then re-enters GAP Limited Discovery Procedure.

When in connection and if notifications are configured, pressing the ALERTSW button triggers a new alert notification (“New mail”) to be sent to the GATT client and adjusts the time of the device by +1 hour. Holding the ALERTSW button for 2-3 seconds triggers a new unread alert status notification (missed call) to be sent to the same GATT client.

See details below for hardware references.

**Table 17. Hardware references**

Platform	SCANSW	CONNLED	ALERTSW
FRDM-KW41Z	SW4	LED3	SW3
QN9080CDK	BUTTON1	D5-RED	BUTTON2
FRDM-KW36	SW2	LED3	SW3
FRDM-KW38	SW2	LED3	SW3

### 5.17.4 Usage

The application can be tested against the Profile Tuning Suite (PTS) black-box testing tool from the Bluetooth SIG.

## 5.18 Wireless UART

This section describes the implemented profiles and services, user interactions, and testing methods for the Wireless UART application.

### 5.18.1 Implemented profile and services

The Wireless UART application implements both the GATT client and server for the custom Wireless UART profile and services.

- Wireless UART Service (UUID: 01ff0100-ba5e-f4ee-5ca1-eb1e5e4b1ce0)
- Battery Service v1.0
- Device Information Service v1.1

The Wireless UART service is a custom service that implements a custom writable ASCII Char characteristic (UUID: 01ff0101-ba5e-f4ee-5ca1-eb1e5e4b1ce0) that holds the character written by the peer device.

The application behaves at first as a GAP central node. It enters GAP Limited Discovery Procedure and searches for other Wireless UART devices to connect. To change the device role to GAP peripheral, use the ROLESW button. The device enters GAP General Discoverable Mode and waits for a GAP central node to connect.

### 5.18.2 Supported platforms

The Wireless UART application is supported by the following platforms:

- FRDM-KW41Z
- USB-KW41Z
- QN9080CDK
- FRDM-KW36
- FRDM-KW38
- USB-KW38

### 5.18.3 User interface

After flashing the board, the device is in idle mode (all LEDs flashing). To start scanning, press the SCANSW button. When in GAP Limited Discovery Procedure or GAP General Discoverable Mode, CONNLED is flashing. When the node connects to a peer device, CONNLED turns solid. To disconnect the node, hold the SCANSW button pressed for 2-3 seconds. The node then re-enters GAP Limited Discovery Procedure.

See details below for hardware references.

**Table 18. Hardware references**

Platform	SCANSW	CONNLED	ROLESW
FRDM-KW41Z	SW4	LED3	SW3
QN9080CDK	BUTTON1	D5-RED	BUTTON2
FRDM-KW36	SW2	LED3	SW3
FRDM-KW38	SW2	LED3	SW3
USB-KW38	SW1 single press	LED2	SW1 double press

### 5.18.4 Usage

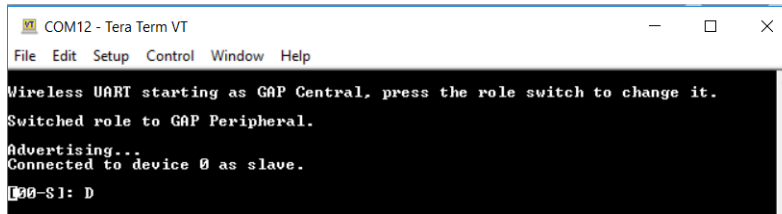
The application is built to work with another supported platform running the same example or with the Wireless UART from the IoT Toolbox application. When testing with two boards do the following.

1. Open a serial port terminal and connect them to the two boards, in the same manner described in section 5.1.3. The start screen is blank after the board is reset.

**NOTE**

The role switch feature is not available on USB-KW38.

2. The application starts as a GAP central. To switch the role to a GAP peripheral, press the role switch. Depending on the role, when pressing the SCANSW, the application starts either scanning or advertising.
3. As soon as the CONNLED turns solid on both devices, the user can start writing in one of the consoles. The text appears on the other terminal, as shown in the figure below.
4. After creating a connection, the role (master or slave) is printed in the console. The role switch can be pressed again before creating a new connection.



**Figure 34. Tera Term – received text on Wireless UART**

When testing with one board and the IoT Toolbox do the following.

1. Open a serial port terminal and connect the board in the same manner described in section 5.1.3. The start screen is blank after the board is reset.
2. Press the role switch button to behave as a GAP peripheral and then press the SCANSW to start advertising. The IoT Toolbox app can then connect. Select UART instead of Console and start typing.

## 5.19 Bluetooth LE Shell

This section describes the functionality, user interactions, and testing methods for the Bluetooth LE Shell Application.

### 5.19.1 Implemented stack features

The Bluetooth LE Shell Application implements a console application that allows the user to interact with a full feature Bluetooth Low Energy stack library. It implements All GAP roles and both GATT client and server. Enabling these roles can be done using shell commands.

### 5.19.2 Implemented profile and services

The application implements a dynamic GATT database. The user can add services at runtime and also erase the database contents. The database is always populated with the GAP and GATT services. These services cannot be erased. The user can dynamically add the following services.

- Heart Rate Service (UUID: 0x180D)
- Battery Service (UUID: 0x180F)
- Device Information Service (UUID: 0x180A)
- Internet Support Profile Service (0x1820)

### 5.19.3 Supported platforms

The Bluetooth LE Shell application is supported by the following platforms:

- FRDM-KW41Z
- USB-KW41Z
- QN9080CDK
- FRDM-KW36
- FRDM-KW38
- USB-KW38

### 5.19.4 User interface

After flashing the board, the device is in idle mode (all LEDs flashing). The interaction with the board is done entirely by using the shell commands via the serial communication terminal.

### 5.19.5 Usage

The application is built to work with any other Bluetooth LE device. To showcase the functionality, two platforms are used in the following setup.

1. Open a serial port terminal and connect them to the two boards, in the same manner described in section 5.1.3. The start screen is displayed after the board is reset. All LEDs are flashing on both devices.
2. Configure one of the devices as a GAP peripheral and a Heart Rate server. Change name to HRS. Start advertising on this device.

```
Kinetis Shell>gap devicename HRS
--> GATTDDB Event: Attribute Written
HRS>gap advdata 1 6
--> GAP Event: Advertising data successfully set.
HRSHRS>gap advdata 8 HRS
--> GAP Event: Advertising data successfully set.
HRS>gap advstart
--> GAP Event: Advertising state changed successfully!
HRS>gattddb addservice 0x180D
--> Heart Rate
```



```
- Heart Rate Measurement Value Handle: 14
- CCCD Handle: 15
- Body Sensor Location Value Handle: 17
- Heart Rate Control Point Value Handle: 19
--> GATTDDB Event: Service Added in database.
```

3. Configure the other device as a GAP central. Change name as Collector. Start scanning and connect to the HRS device by selecting the corresponding device index. In the example below, the HRS device is device number 2.

```
Kinetis Shell>gap devicename Collector
--> GATTDDB Event: Attribute Written
Collector>gap scanstart
--> GAP Event: Scan started.
Collector>
--> GAP Event: Found device 0 : 880F102F500E 0 dBm
--> GAP Event: Found device 1 : NXP_CSCS 00049F000006 0 dBm
--> GAP Event: Found device 2 : HRS 00049F0000FF 0 dBm
Collector>gap connect 2
--> GAP Event: Scan stopped.
Collector>
--> GAP Event: Connected to peer 0
```

4. Optionally, the devices can be paired. On the collector initiate the pairing.

```
Collector>gap pair 0
--> Pairing...
--> GAP Event: Passkey is 792910
```

5. On the HRS, enter the pin displayed by the collector, as shown below. If the `leSecureConnectionSupported` member of `gPairingParameters`, from `app_config.c` is enabled, then the PIN is not required.

```
HRS>
--> GAP Event: PIN required
HRS>gap enterpin 0 792910
--> GAP Event: Device Paired.
```

6. On the Collector, start service discovery. The device discovers the GAP, GATT and Heart Rate services.

```
Collector>gatt discover 0 -all
--> Discovered primary services: 3
--> Generic Access Start Handle: 1 End Handle: 7
- Device Name Value Handle: 3
- Appearance Value Handle: 5
- Peripheral Preferred Connection Parameters Value Handle: 7
--> Generic Attribute Start Handle: 8 End Handle: 11
- Service Changed Value Handle: 10
- Client Characteristic Configuration Descriptor Handle: 11
--> Heart Rate Start Handle: 12 End Handle: 19
- Heart Rate Measurement Value Handle: 14
- Client Characteristic Configuration Descriptor Handle: 15
- Body Sensor Location Value Handle: 17
- Heart Rate Control Point Value Handle: 19
```

7. Configure the HRS to send notifications by writing the CCCD from the Collector. Send a GATT write command with value 1 to the CCCD handle discovered, 15.

```
Collector>gatt write 0 15 0x0001
```

- Send heart rate measurement notifications from the HRS device by using the value handle obtained after adding the service in the previous step.

```
HRS>gatt notify 0 14
```

- A notification appears on the Collector console.

```
Collector>
--> GATT Event: Received Notification
Handle: 14
Value: B400
```

### 5.19.5.1 Extended Advertising (KW37 platforms)

Use the Bluetooth LE Shell application to exercise the advertising extension features:

On the GAP Peripheral device:

- Configure the extended advertising parameters. In the below example the advertising type is set to connectable and includes TX power and the primary phy is set to Coded PHY.
- Configure the extended advertising data. To send large data payload, pass the command “gap extadvdata” with no parameter and the default data is added. The length is configurable at compile time through SHELL\_EXT\_ADV\_DATA\_SIZE and the data pattern is SHELL\_EXT\_ADV\_DATA\_PATTERN. In the below example custom data is added.
- Start extended advertising.

```
BLE Shell>gap extadvcfg -type 65 -phy1 3
--> GAP Event: Extended Advertising parameters successfully set.

BLE Shell>gap extadvdata 8 test_ext_adv_data
--> GAP Event: Extended Advertising data successfully set.

BLE Shell>gap extadvstart
--> GAP Event: Advertising state changed successfully!
```

On the GAP Central device:

- Set the scanning parameters. The scanning PHY is set to match the advertising PHY, in this case Coded PHY.
- Start scanning and filter duplicates.

```
BLE Shell>gap scancfg -phy 4

BLE Shell>gap scanstart filter
-> GAP Event: Scan started.

BLE Shell>
--> GAP Event: Found device 0 : 0060374E2B8A -22 dBm
Advertising Extended Data:
test_ext_adv_data
```

- Set the connection initiating PHYs corresponding to the primary PHY on which the advertising is performed.

## 7. Connect to the desired device in the scanned devices list.

```
BLE Shell>gap connectcfg -phy 4

--> Connection Parameters:
--> Connection Interval: 200 ms
--> Connection Latency: 0
--> Supervision Timeout: 32000 ms
--> Connecting PHYs: Coded

BLE Shell>gap connect 0
-> GAP Event: Scan stopped.

BLE Shell>
--> GAP Event: Connected
```

### 5.19.5.2 Periodic Advertising

Exercise periodic advertising features using the BLE Shell application as follows:

On the GAP Peripheral device:

#### 1. Set the periodic advertising parameters and data. Start extended advertising and periodic advertising.

```
BLE Shell>gap periodiccfg
--> Periodic Advertising Parameters:
--> Periodic Advertising Handle: 1
--> Periodic Advertising Interval: 2000 ms

--> GAP Event: Periodic Advertising parameters successfully set.

BLE Shell>gap periodicdata 8 periodicdata
--> GAP Event: Periodic Advertising data successfully set.

BLE Shell>gap extadvstart
--> GAP Event: Advertising state changed successfully!

BLE Shell>gap periodicstart
--> GAP Event: Advertising state changed successfully!

BLE Shell>gap address
--> GAP Event: Public Address Read:0060374E2B8A
```

On the GAP Central device:

#### 2. Start scanning. Create periodic advertising sync with the advertiser:

```
BLE Shell>gap periodicsync -peer 0060374E2B8A -type 0
BLE Shell>gap scanstart
BLE Shell>
--> GAP Event: Periodic Advertising Sync Established
--> GAP Event: Periodic Device Scanned
--> GAP Event: Periodic Device Scanned
--> GAP Event: Periodic Device Scanned
```

### 5.19.5.3 RSSI Monitor

RSSI Monitor is an application that allows monitoring the RSSI of a remote peer on advertising or connection channel. The GAP Peripheral device can modify the output Tx power on both advertising and connection channels. For platforms that do not support Bluetooth LE Optional features, use the legacy advertising and default phy for this feature.

On GAP Peripheral device

1. Set the primary advertising PHY to Coded PHY. Start advertising and read the address. Set the Tx power in dbm to a value less than 20 dbm.

#### NOTE

For FRDM-KW38 the accepted range of values is [-127, 20]. For the Tx power to dBm lookup table see platform's datasheet (Section 4.3, Table 6, for FRDM-KW38).

```
BLE Shell>gap extadvcfg -phy1 3
--> GAP Event: Extended Advertising parameters successfully set.

BLE Shell>gap extadvstart
--> GAP Event: Advertising state changed successfully!

BLE Shell>gap address
--> GAP Event: Public Address Read:00603701A751

BLE Shell>gap txpower adv 0
--> GAP Event: Success!
```

On GAP Central device:

2. Set the scanning PHY to Coded PHY. Start monitoring the RSSI on advertising channel. Scanning is started automatically, if not previously enabled.

```
BLE Shell>gap scancfg -phy 4
BLE Shell>gap rssimonitor 00603701A751
--> Reading RSSI on advertising channel:
-> GAP Event: Scan started.

BLE Shell>
RSSI: -31 dBm
RSSI: -30 dBm
RSSI: -30 dBm
RSSI: -30 dBm
```

In the below example it is monitored the RSSI on a connection channel.

On GAP Peripheral, start advertising in connectable mode on Coded PHY and adjust the TX power level.

```
BLE Shell>gap extadvcfg -type 65 -phy1 3
--> GAP Event: Extended Advertising parameters successfully set.

BLE Shell>gap extadvdata 8 rssimonitortest
--> GAP Event: Extended Advertising data successfully set.

BLE Shell>gap extadvstart
--> GAP Event: Advertising state changed successfully!

BLE Shell>
--> GAP Event: Connected to peer 0

BLE Shell>
```

```
--> GAP Event: Advertising stopped!  
  
BLE Shell>gap txpower conn 10  
--> GAP Event: Success!
```

On GAP Central device, start scanning on Coded PHY, update the connection PHY also to Coded PHY, and connect to remote device and monitor continuously the RSSI on the connection channel.

```
BLE Shell>gap scancfg -phy 4  
BLE Shell>gap connectcfg -phy 4  
  
--> Connection Parameters:  
--> Connection Interval: 200 ms  
--> Connection Latency: 0  
--> Supervision Timeout: 32000 ms  
--> Connecting PHYs: Coded  
  
BLE Shell>gap scanstart filter  
  
-> GAP Event: Scan started.  
  
BLE Shell>  
--> GAP Event: Found device 0 : 0060374E2B8A -30 dBm  
Advertising Extended Data:  
rssiannotortest  
BLE Shell>gap connect 0  
  
-> GAP Event: Scan stopped.  
  
BLE Shell>  
--> GAP Event: Connected to peer 0  
  
BLE Shell>gap rssiannotor 0 -c  
  
--> Reading RSSI from connected device:  
RSSI: -29 dBm  
RSSI: -29 dBm  
RSSI: -29 dBm  
RSSI: -29 dBm  
RSSI: -29 dBm
```

3. Update the PHY preference and continue monitoring the RSSI. For Coded PHY the coding scheme can be configured between S2 and S8 (500kb/s and 125kb/s).

```
BLE Shell>gap phy 0 -tx 4 -rx 4 -o 1  
  
BLE Shell>  
--> GAP Event: Phy update complete!  
--> TxPhy: Coded  
--> RxPhy: Coded  
  
BLE Shell>gap phy 0 -tx 4 -rx 4 -o 2  
  
BLE Shell>  
--> GAP Event: Phy update complete!  
--> TxPhy: Coded  
--> RxPhy: Coded
```

```

BLE Shell>gap phy 0 -tx 2 -rx 2

BLE Shell>
--> GAP Event: Phy update complete!
--> TxPhy: 2M
--> RxPhy: 2M

BLE Shell>gap rssimonitor 0 -c

--> Reading RSSI from connected device:
RSSI: -21 dBm
RSSI: -22 dBm

```

### 5.19.6 Throughput feature

The Bluetooth LE Shell application also has a throughput test feature which can be used to test different combinations of connection interval, payload size, and packet count to determine the best data-rate.

This feature requires two devices:

- GAP Peripheral: transmits the test packets
- GAP Central: receives the packets and display a report

All throughput related commands are grouped under the “thruput” keyword:

- thrput start tx: configures the device as a GAP Peripheral and start advertising. Once the receiving device is connected, the packet transmission will start. The packet size and count can also be specified (-s <size\_value> -c <count\_value>).
- thrput start rx: configures the device as a GAP Central and will start scanning. Once a transmitter device is found, it will connect to it and wait for the test to begin. The connection interval can also be configured (-ci <value>).
- thrput stop: will stop the test and disconnect the devices.

Once a connection has been established between the devices and initial throughput test end, one can start a new throughput transmission test with a new set of parameters (packet size / count).

The receiving device will generate the report if no packets were received for more 3 consecutive connection events.

The default config of the throughput test is the following:

Packet count: 1000

Payload size: 20 bytes

Connection interval (min,max): 160, 160 (200ms)

Example of a test report:

```

*****
***** TEST REPORT *****
*****
Packets received: 1000
Total bytes: 244000
Receive duration: 4162 ms
Average bitrate: 469 kbps
*****
***** END OF REPORT *****
*****

```

## 5.20 Over the Air Programming (OTAP)

This section describes the implemented profiles and services, user interactions, and testing methods for the Bluetooth LE OTAP application.

### 5.20.1 Implemented profile and services

The Bluetooth LE OTAP applications implement the GATT client and server for the custom Bluetooth LE OTAP profile and service.

- Bluetooth LE OTAP Service (UUID: 01ff5550-ba5e-f4ee-5ca1-eb1e5e4b1ce0)

The Bluetooth LE OTAP Service is a custom service which has 2 characteristics.

- OTAP Control Point Characteristic (UUID: 01ff5551-ba5e-f4ee-5ca1-eb1e5e4b1ce0). This characteristic can be written and indicated to exchange OTAP Commands between the OTAP Server and the OTAP Client. Data chunks are not transferred using this characteristic.
- OTAP Data Characteristic (UUID: 01ff5552-ba5e-f4ee-5ca1-eb1e5e4b1ce0). This characteristic can be written without response by the OTAP Server to transfer image file data chunks to the OTAP Client only when an image block transfer is requested via the ATT transfer method. Data chunks can also be transferred via the L2CAP credit based PSM channels method.

The demo runs using 4 applications: an OTAP Client embedded application, an OTAP Bootloader embedded application, an OTAP Server embedded application (for QN9090 the bootloader is not required, in case of K32W061 a Secondary Stage Bootloader - SSBL is required), and an OTAP Server PC application. The OTAP Client embedded application has two versions, an ATT version and a L2CAP version each using a different transfer method. The OTAP Bootloader (SSBL for the K32W061) and OTAP Client applications must be flashed on the same device (the program region does not overlap).

The embedded OTAP Server application is a GAP Central application which scans for devices advertising the Bluetooth LE OTAP service. After it finds one, it connects to it and configures the OTAP Control Point CCC Descriptor to receive ATT Indications from the device then it waits for OTAP commands from this device.

Once commands start arriving from the OTAP Client via ATT Indications the OTAP Server relays them via serial interface to a PC application which responds. The responses are then sent back to the OTAP Client by writing the OTAP Control Point Characteristic. The embedded OTAP Server application effectively acts as a relay between the OTAP Client to which the image is sent over the air and the OTAP Server PC application which has an OTAP image file constructed using a binary .srec image or a .bin image.

The OTAP Client is a GAP Peripheral which advertises the Bluetooth LE OTAP Service and waits for a connection from an OTAP Server. After an OTAP Server connects, the OTAP Client waits for it to write the OTAP Control Point CCCD and then starts sending commands via ATT Indications. If the OTAP Client is configured to ask the data transfer via the L2CAP CoC PSM, it registers and tries to connect a predetermined L2CAP PSM before sending any commands to the OTAP Server.

### 5.20.2 Supported platforms

The OTAP application is supported by the following platforms:

- FRDM-KW41Z
- USB-KW41Z
- QN9080CDK
- FRDM-KW36
- FRDM-KW38
- USB-KW38 (OTA Server only)

### 5.20.3 User interface

After flashing two boards with the OTAP Server and OTAP Client applications respectively, the devices are in idle mode (all LEDs flashing). To start advertising, press the ADVSW button on the OTAP Client. To start scanning, press the SCANSW button on

the OTAP Server. After the two devices connect and start exchanging commands. CONNLED becomes solid on the OTAP Server and on the OTAP Client.

After the embedded applications are flashed to the boards the OTAP Server PC application must be started. The application creates an OTAP image file using the provided executable .srec or .bin file, connects to the embedded OTAP Server via the configured serial interface and waits for commands. The application shows details about the creation of the image file and allows the configuration of the OTAP upgrade image file header. A log view is present where the interactions between the OTAP Client and the OTAP Server are shown.

**Table 19. Hardware references**

Platform	ADVSW	SCANSW	CONNLED
FRDM-KW41Z	SW4	SW4	LED3
QN9080CDK	BUTTON1	BUTTON1	D5-RED
FRDM-KW36	SW2	SW2	LED3
FRDM-KW38	SW2	SW2	LED3
USB-KW38	–	SW1	LED2

#### 5.20.4 Usage with Test Tool for Connectivity products

This is a list of requirements.

- Test Tool for Connectivity Products 12.5.0 or newer – Test Tool on [www.nxp.com](http://www.nxp.com)
- Serial COM port drivers – these are board-specific

These are the steps to run the application.

1. Flash the OTAP Server onto a supported platform, and the OTAP Bootloader and the OTAP Client to another supported platform. Make sure the board running the OTAP Server is connected to your PC and your PC has appropriate drivers for the USB to serial device on that board.

#### NOTE

The OTAP Bootloader must be programmed separately into the MCU, before programming the OTAP Client application.

2. Create the application to send over the air. The executable must be provided in the .srec or .bin format. The .srec format executable can be obtained by using the IAR Output Converter and setting the output format to Motorola as shown below. The created .srec application image must be offset to begin after the Bootloader region (not applicable for a QN9090 device). To offset the application copy the settings from the Linker->Config tab of the otap\_client\_i2cap\_credit or otap\_client\_att example applications project properties. For more details, see the *Bluetooth LE Application Developer's Guide* (BLEADG).



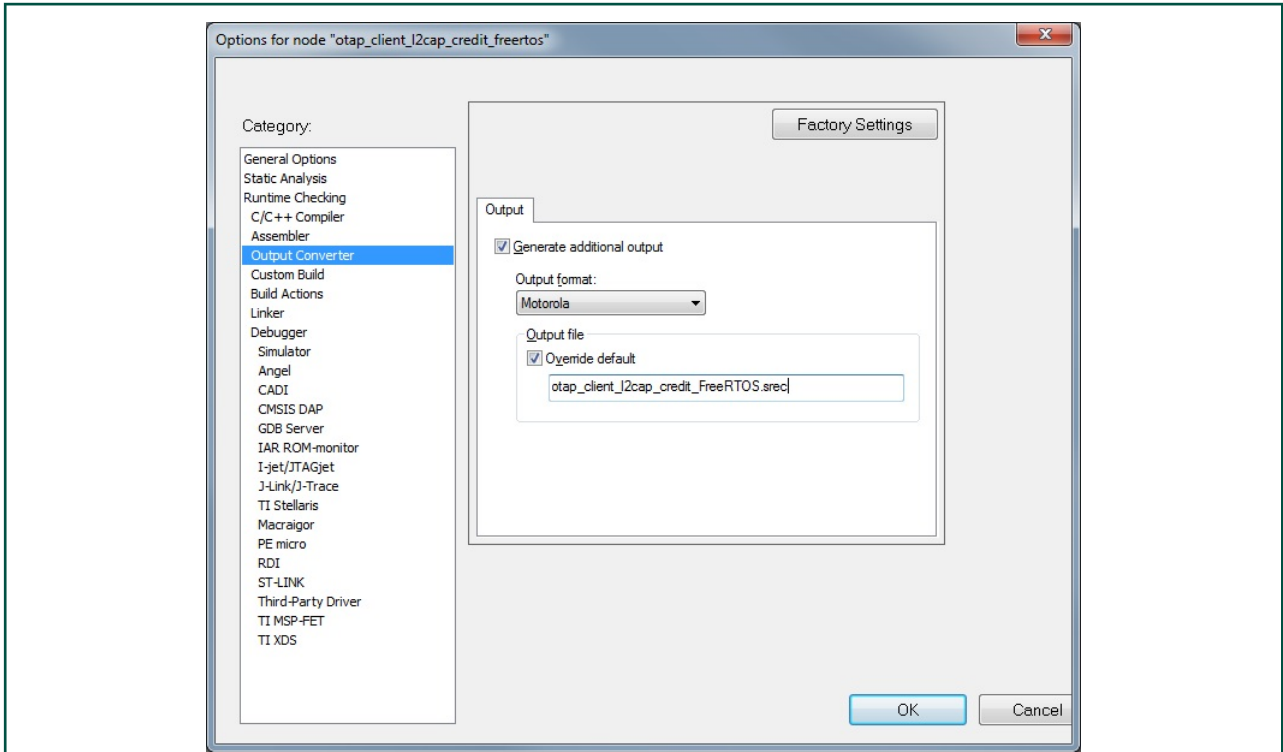


Figure 35. IAR Output Converter Dialog - .srec output

- To obtain a .bin file from IAR select the Raw binary option in the IAR Output Converter as seen in the figure below.

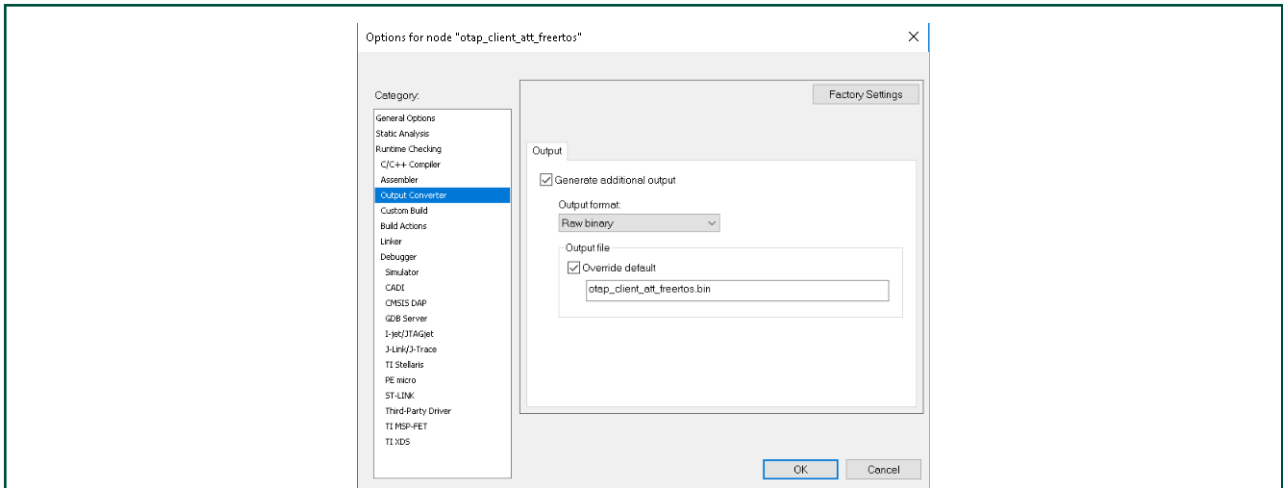


Figure 36. IAR .bin file output converter

- To obtain a .bin file from MCUXpresso IDE go to the Project properties -> Settings -> Build steps window and press the Edit button for the Post-build steps. A Post-build steps window will show up in which the following command must be added or uncommented (by removing the '#' character at the beginning) if it is already there: `arm-none-eabi-objcopy -v -O binary "${BuildArtifactFileName}" "${BuildArtifactFileName}.bin"`. To obtain a .srec (.s19) file add or uncomment the following post-build command in the same window: `arm-none-eabi-objcopy -v -O srec "${BuildArtifactFileName}" "${BuildArtifactFileName}.s19"`. This window, is shown in the figure below.

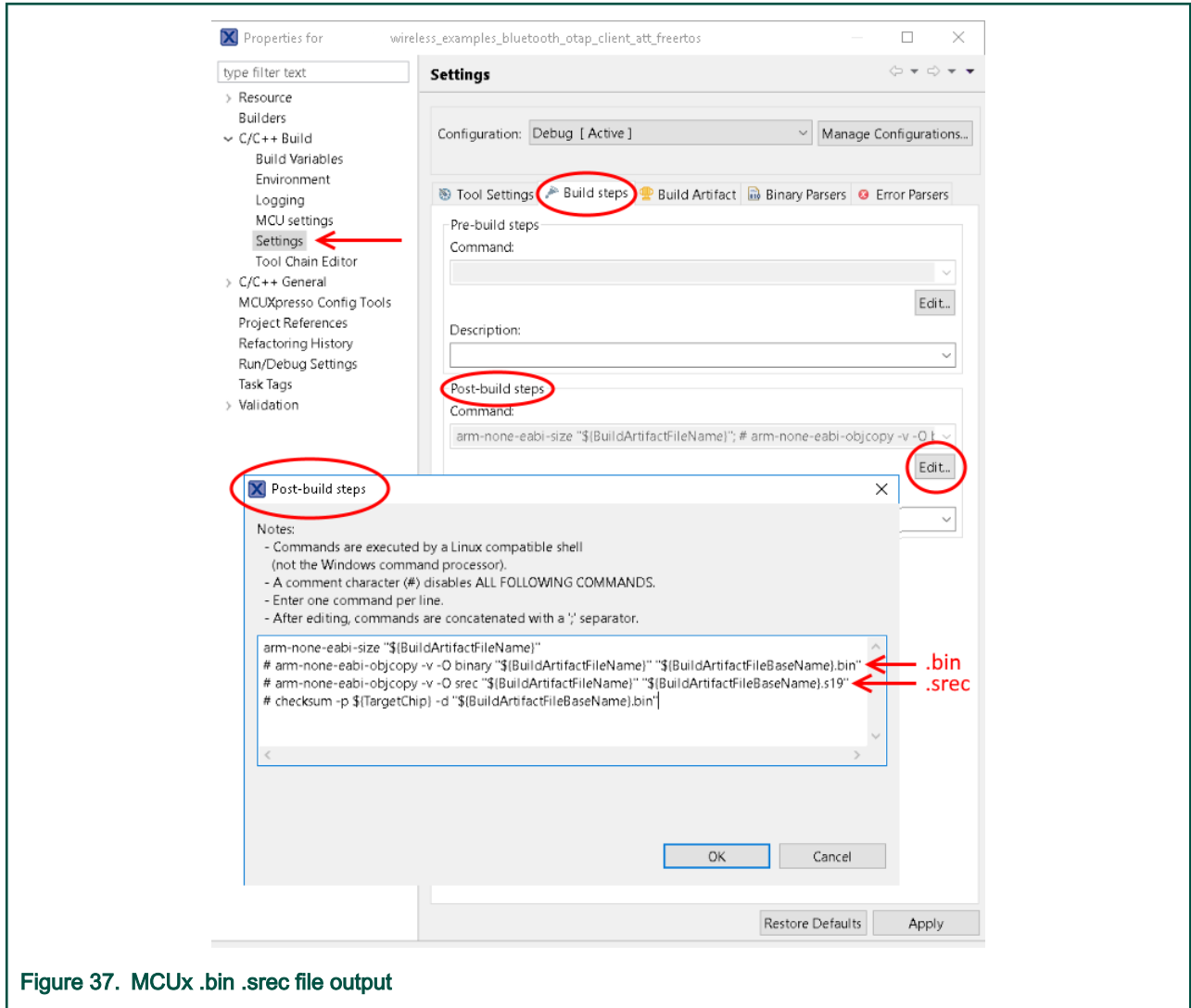


Figure 37. MCUx .bin .srec file output

5. Start the Test Tool application. If you have the proper drivers installed and the OTAP Server board is connected to the PC, then its corresponding serial port shows up in the USB/UART Active devices view on the Start Page of the Test Tool software as shown below.



Figure 38. Test Tool Start Page - OTAP Server USB/UART

6. Go to the OTA Updates menu in Test Tool and choose the OTAP BLE option to open the Bluetooth LE over-the-air update window.



Figure 39. Test Tool OTAP Bluetooth LE

7. Load the image file into the application and configure the image file header and start the OTAP Server
  - In the Image File Information box of the OTAP Bluetooth LE application from Test Tool press the “Browse” button and go to the .srec or .bin file containing the image to be sent to the OTAP Client. The .srec or .bin file and the OTA Header configuration options from the same box is used by the application to build the OTAP Image File which is sent over-the-air. The default values of the OTA Header configuration work out of the box for the OTAP demo applications. For details about these configuration options, see the Bluetooth LE Application Developer’s Guide document (BLEADG). After the .srec or .bin file is chosen a pop-up window asks to choose the target processor (this is used to correctly configure the Sector Bitmap sub-element of the OTAP Image File). Choose the correct processor and press “OK”. The "Image contains bootloader" tick-box only needs to be selected for KW41Z devices.

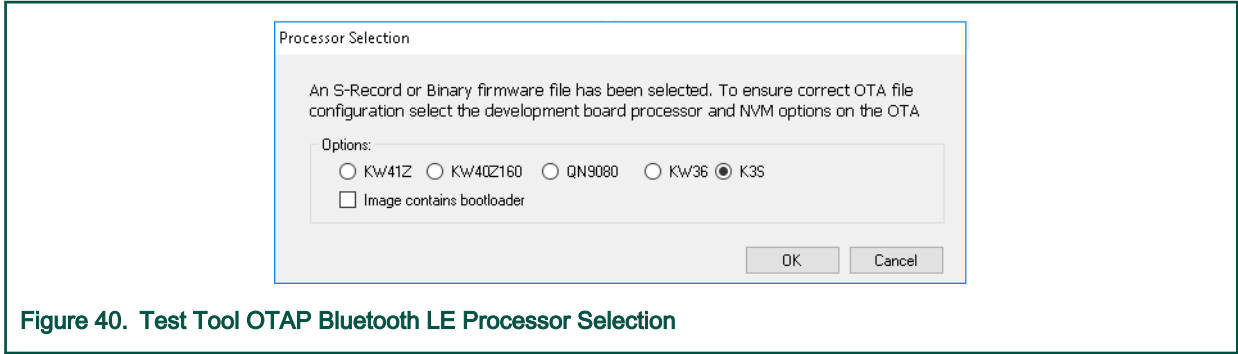


Figure 40. Test Tool OTAP Bluetooth LE Processor Selection

- After the image is loaded go to the OTA Server Image Loading box, select the correct COM Port for the OTAP Server board and the default baudrate of 115200 and press the “Connect to OTAP Server Device” button. If the connection is successful then the Message Log shows this. If the image is loaded before connecting to the OATP Server COM Port, then the application’s OTAP Server starts automatically. If the connection to the COM Port is established before the image is loaded, then the “Start OTAP” button needs to be pressed to start the application’s OTAP Server. For details, see the annotated screenshot below.

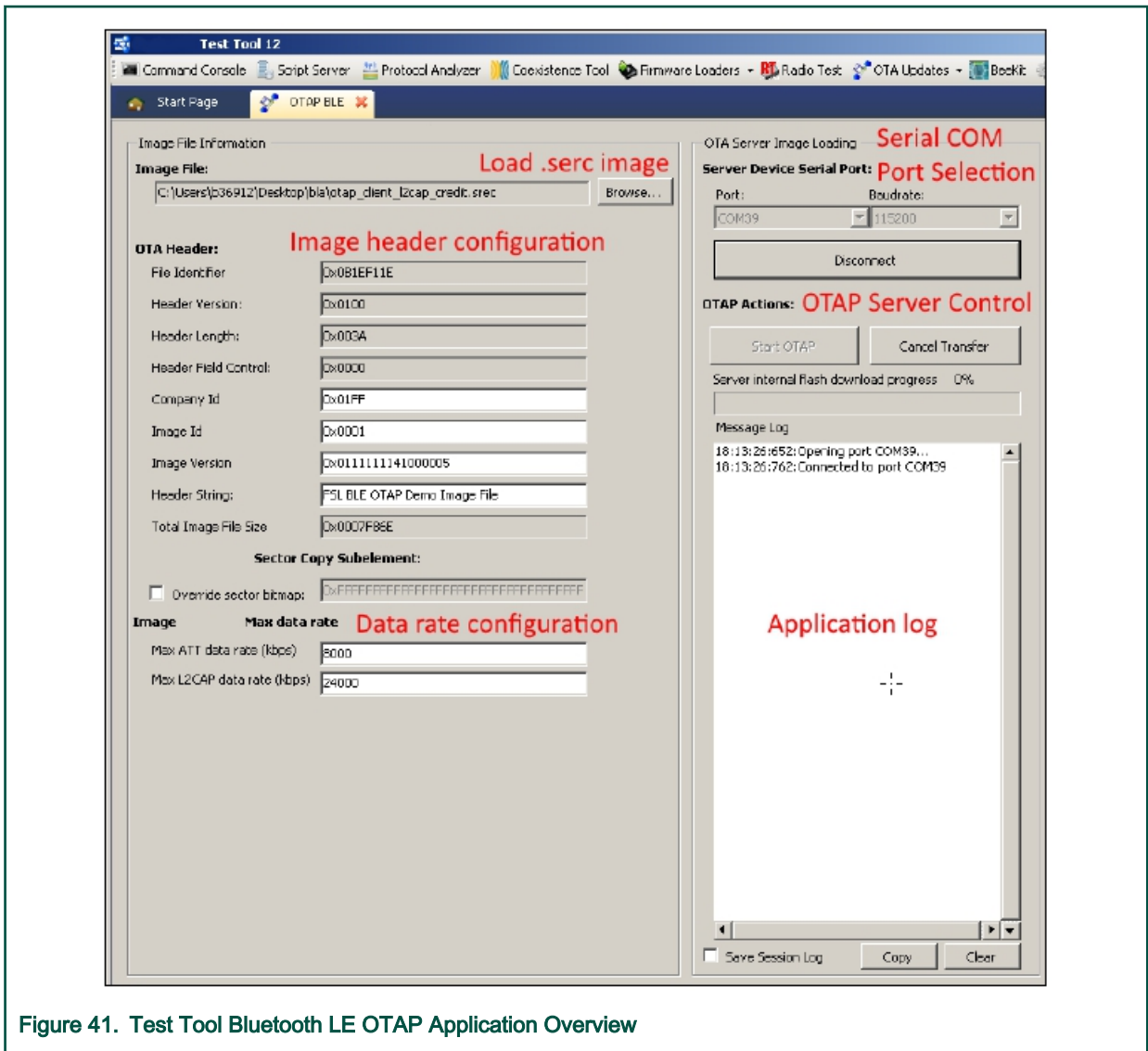


Figure 41. Test Tool Bluetooth LE OTAP Application Overview

- Before starting the image transfer process, the data rate must be configured for each transfer method (ATT or L2CAP CoC). The image chunks of a block are sent over the serial interface and over-the-air without waiting for confirmation and if the data rate is not configured correctly errors can appear in the transfer process which can slow it down significantly. The optimal data rate depends on multiple factors like: distance between boards, type of antenna, performance of the RF circuitry between the radio and antenna, type and level of noise in the environment, speed of the storage medium in which the image is saved on the OTAP Client, serial driver delay between PC and OTAP Server board and other factors. If the data rate is too high, then the OTAP Client receives a new chunk before it can process the previous one and it sends an “Unexpected Chunk Sequence Number” error and restart the transfer of the current block from where it left off. If the channel is too noisy the transmitter can be flooded and some chunks may not reach the client triggering a similar type of error. The default data rate values should work for most configurations.
8. Start the embedded applications by pressing ADVSW first on the OTAP Client and then on the OTAP Server. The transfer progress and transfer-related messages and/or errors are shown in the application window. The duration of the transfer depends on the size of the image and the chosen data rate and transfer method.

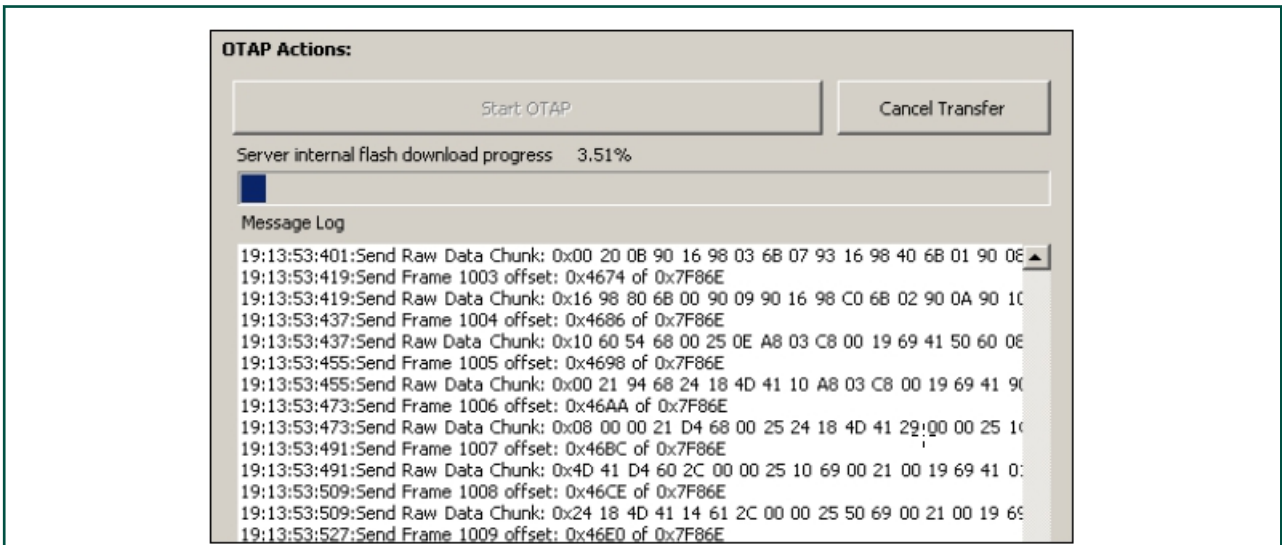


Figure 42. Test Tool OTAP Bluetooth LE Image Transfer in Progress

9. After all the blocks are sent the OTAP Client send an Image Transfer Complete command to the OTAP Server. When this command is received by the PC Application, it displays a “Sent Image with Success” message in the log window.

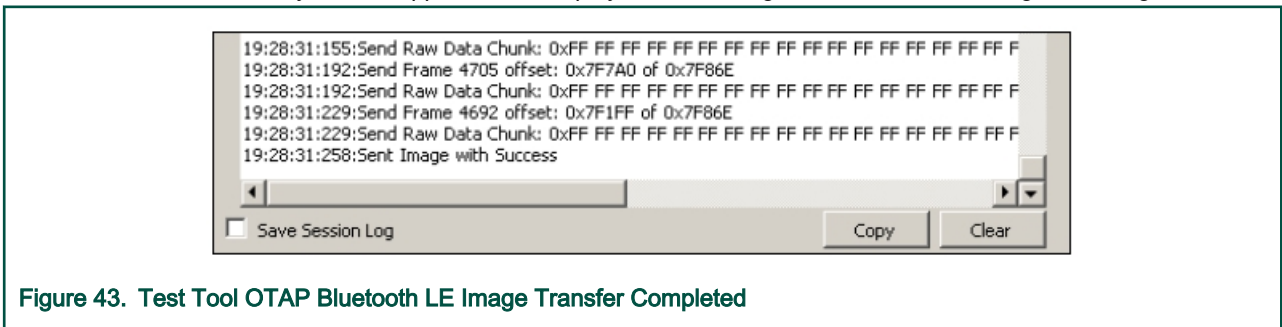


Figure 43. Test Tool OTAP Bluetooth LE Image Transfer Completed

10. After the image transfer is complete the OTAP Client triggers the bootloader and resets the MCU. The bootloader takes about 30 seconds to flash the image on the board. After this time passes the MCU resets again and runs the new image.

**NOTE**

For QN9090 or K32W061, [#unique\\_120](#) explains what will happen once the image transfer is completed.

## 5.20.5 Usage with IoT Toolbox

This is the list of requirements.

- Mobile device running Android platform or iOS with hardware and software supporting Bluetooth 4.0 and later
- Kinetis Bluetooth LE Toolbox application – download from the specific application store for your device

To run the application do the following.

1. Flash the OTAP Client ATT and the OTAP Bootloader applications to a supported platform. The Kinetis Bluetooth LE Toolbox only supports the ATT OTAP Client
2. Create the application to send over the air in .srec format. Follow the instructions from the previous section on how to do this. Remember to include the Bootloader in its appropriate section in your application.
3. Start the Kinetis Bluetooth LE Toolbox application on your mobile device and start the OTAP Tool. The application starts scanning.
4. Press ADVSW on the board to start Advertising on the embedded OTAP Client application. The device should show up in the list of scanned devices. Touch the device in the scan list to connect to and the application performs service discovery and displays some information shown in the figures below.

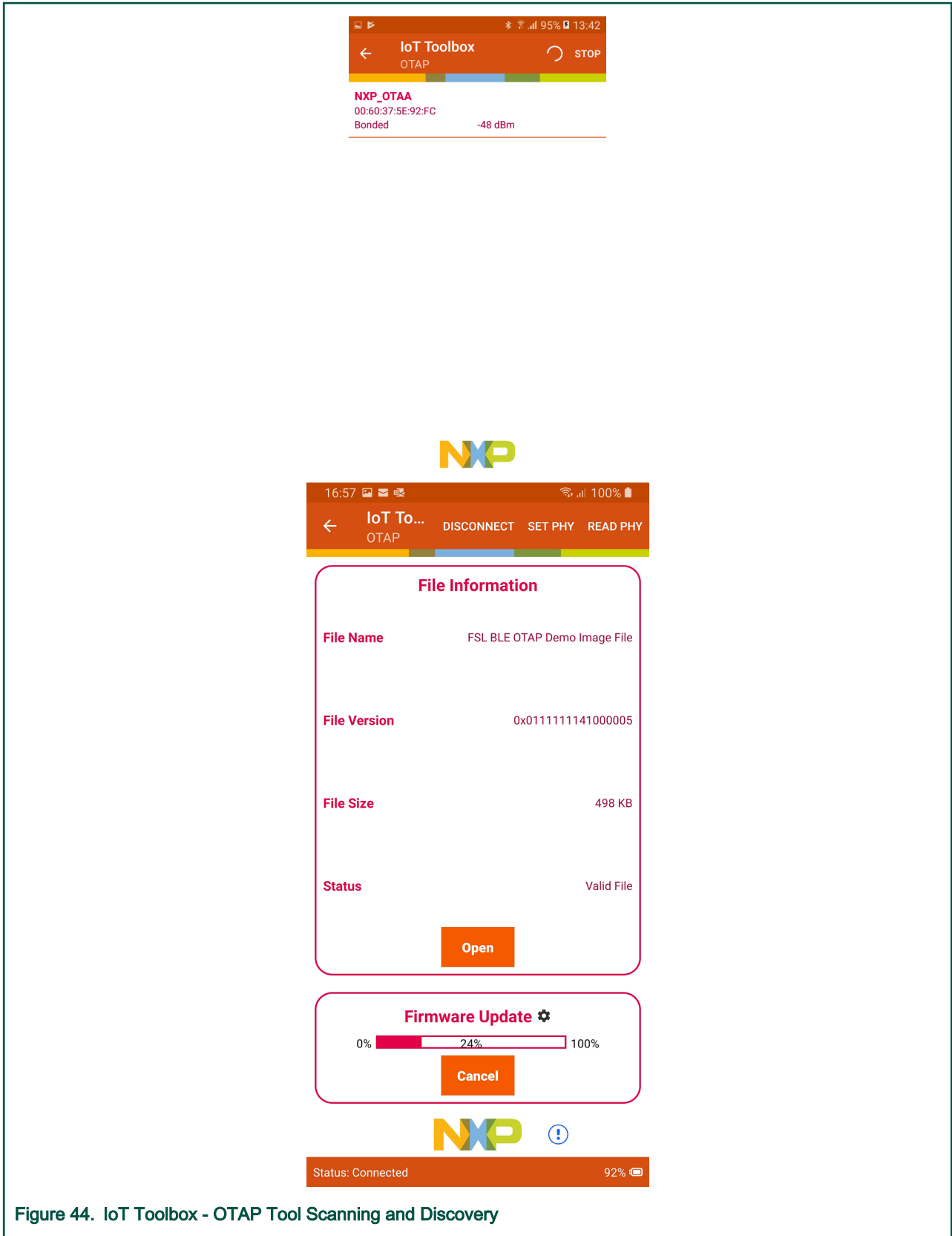


Figure 44. IoT Toolbox - OTAP Tool Scanning and Discovery

5. Press the “Open” button and load the .srec file to be sent over-the-air. Once the file is loaded some information about it is displayed. Press the “Upload” button to start the image transfer process. A progress bar is shown while the image

transfer is ongoing. The successful transfer is signaled by the progress bar reaching 100%. This is shown in the figures below.

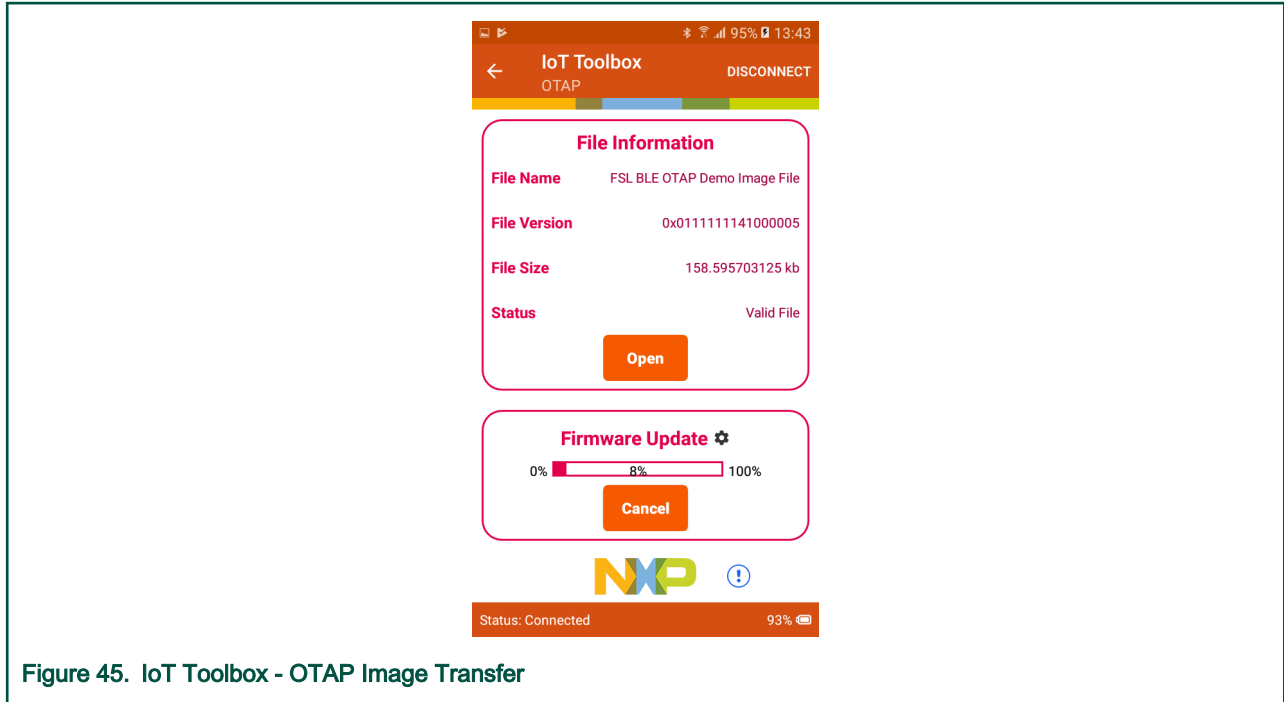


Figure 45. IoT Toolbox - OTAP Image Transfer

- After the image transfer is complete the OTAP Client triggers the bootloader and resets the MCU. The bootloader takes about 30 seconds to flash the image on the board. After this time passes the MCU resets again and runs the new image.

## 5.21 Hybrid (Dual-Mode) Bluetooth Low Energy Heart Rate Sensor and IEEE 802.15.4 Coordinator Demo Application (KW41 Platform)

This section describes the implemented profiles and services, user interactions, and testing methods for the Hybrid Bluetooth LE Heart Rate Sensor/IEEE<sup>®</sup> 802.15.4 Coordinator demo application. This section only applies to KW41Z, the only platform offering support for 802.15.4.

### 5.21.1 Implemented profile and services

The Bluetooth LE Heart Rate Sensor implements a GATT server and the following profile and services.

- Heart Rate Profile v1.0
- Heart Rate Service v1.0
- Battery Service v1.0
- Device Information Service v1.1

The application behaves as a GAP peripheral node. It enters GAP General Discoverable Mode and waits for a GAP central node to connect. When the collector configures notifications, the sensor starts sending heart rate measurements every second.

The IEEE 802.15.4 part of the application is an IEEE 802.15.4 Coordinator running in background, in parallel with the Bluetooth LE sensor application.

### 5.21.2 Supported platforms

The Hybrid (Dual-Mode) Bluetooth Low Energy Heart Rate Sensor and IEEE 802.15.4 Coordinator application is supported by the following platforms:



- FRDM-KW41Z
- USB-KW41Z

### 5.21.3 User interface

The IEEE 802.15.4 thread starts automatically at device startup in the role of an IEEE 802.15.4 Coordinator. It chooses the channel based on Clear Channel Assessment (CCA). No user interaction is needed.

The Bluetooth LE Heart Rate Sensor user interface is identical with the standalone one. After flashing the board, the sensor is in idle mode (all LEDs flashing). To start advertising, press the ADVSW button. When in GAP Discoverable Mode, the CONNLED is flashing. When the central node connects to the peripheral, CONNLED turns solid. To disconnect the node, hold the ADVSW button pressed for 2-3 seconds. The node then re-enters the GAP Discoverable Mode.

See details below for hardware references.

**Table 20. Hardware references**

Platform	ADVSW	CONNLED
FRDM-KW41Z	SW4	LED3

### 5.21.4 Testing method

The IEEE 802.15.4 Coordinator is basically a “MyWirelessApp Coordinator” demo (for more information check the 802154MPDAUG document), and responds to requests from other IEEE 802.15.4 devices (Active Scan, Association, Data Requests). Any IEEE 802.15.4 sniffer can be used to observe the over-the-air activity on the channel that the coordinator started.

At the same time, the Bluetooth LE Heart Rate Sensor part of the application runs independently based on user interaction. The Heart Rate Sensor can be connected to any Bluetooth Smart Ready product available on the market. To make the Heart Rate Sensor visible, press the ADVSW button to start sending advertisements, which causes CONNLED to start flashing. The sensor name “NXP\_HYBRID\_HRS” shows on the device when its scanning is active. A solid CONNLED indicates a successful connection between the 2 devices. Hold the ADVSW button pressed for 2-3 seconds anytime to get the sensor to initiate disconnect.

If configured, the sensor notifies the application with heart rate measurements every second. Also, the battery level and various device information is exposed for reading. The Kinetis Bluetooth LE Toolbox can be used to showcase the Heart Rate profile functionality.

## 5.22 Relay Proxy

This section describes the functionality, user interactions, and testing methods for the Bluetooth LE Relay Proxy Application.

### 5.22.1 Implemented stack features

The Bluetooth LE Relay Proxy implements a GAP dual-role application that allows 2 simultaneous connections, one with a GAP Peripheral and the other with a GAP Central device. It also supports both GATT client and server. When the two connections are active, the device will relay data from the GAP Peripheral to the GAP Central and vice-versa.

### 5.22.2 Implemented services

The device will discover services on the GAP Peripheral and clone the database locally by using the dynamic GATT database feature. The application acts afterwards as a proxy for that device and will respond to service discovery queries made by the remote GAP Central device. The application supports all services defined by the Bluetooth SIG. Connection with the GAP Peripheral that contains the database to be cloned is made based on the specified service in the following macro from app.h: gAppProxySelectService\_d. By default, the application will check for GAP Peripherals that support the Heart Rate Service (0x180D).

### 5.22.3 Supported platforms

The Relay Proxy application is supported by the following platforms:

- FRDM-KW41Z
- USB-KW41Z
- QN9080CDK
- FRDM-KW36
- FRDM-KW38

### 5.22.4 User interface

After flashing the board, the sensor is in idle mode (all LEDs flashing). To start scanning, press the SCANSW button. When in GAP Limited Discovery Procedure, CONNLED is flashing. When the central node connects to the peripheral, CONNLED turns solid. After service discovery is done and the dynamic GATT database is created, the application enters GAP Discoverable Mode and CONNLED starts flashing. When the central node connects to the peripheral, CONNLED turns back solid. To disconnect the node from all connections, hold the SCANSW button pressed for 2-3 seconds.

See details below for hardware references.

**Table 21. Hardware references**

Platform	SCANSW	CONNLED
FRDM-KW41Z	SW4	LED3
QN9080CDK	BUTTON1	D5-RED
FRDM-KW36	SW2	LED3
FRDM-KW38	SW2	LED3

### 5.22.5 Usage

The application is built to work with any other two devices that implement a GAP Peripheral and a GAP Central. The out of the box example describes the Bluetooth LE Relay Proxy connecting with a Heart Rate Sensor application from the same SDK and a smartphone or a tablet running the Kinetis Bluetooth LE Toolbox application.

1. Download a Heart Rate Sensor application on a supported platform and start it as described in the previous chapter.
2. Press SCANSW on the Bluetooth LE Relay Proxy application. The node will connect with the Heart Rate Sensor and do Service Discovery. CONNLED is solid. After the database is recreated locally, the node will start advertising with the same data as the original device. CONNLED starts flashing.
3. Start Kinetis Bluetooth LE Toolbox and connect to the Bluetooth LE Relay Proxy application as with a normal Heart Rate Sensor application.
4. All GATT Read and Write operations made by the Kinetis Bluetooth LE Toolbox are relayed by the node to the initial Heart Rate Sensor. All GATT Notifications are relayed to the Kinetis Bluetooth LE Toolbox application.

## 5.23 Wireless Power Transfer System Receiving Unit and Transmitting Unit

This section describes the implemented profiles and services, user interactions, and testing methods for the wireless power transfer receiving unit (WPT PRU) and transmitting unit (WPT PTU).

### 5.23.1 Implemented profile and services

The WPT PRU and PTU applications implement a GATT server and a GATT client for the following profile and service:

- A4WP Wireless Power Transfer System specification v1.3
- Wireless Power Transfer (WPT) Service v1.3

The WPT PRU application behaves as a GAP peripheral node. It enters GAP Limited Discoverable Mode using as advertising data the WPT service UUID (0xFFFE) and the ATT database handle corresponding to the service declaration.

The WPT PTU application behaves as a GAP central node. It scans for PRU devices by looking for the WPT Service UUID, it connects to a PRU device, performs a procedure known as device registration and allows or disallows the device to start charging. The device registration procedure should take less than 500 milliseconds and for this reason the profile uses security mode 1, level 1 and no service discovery is performed. All handles are computed using the offset handle found in the advertising data. Device charging state and other measurable values are simulated in software using random numbers (except for device temperature included in the dynamic parameters).

Both application use pairing with bonding by default. Both applications require a terminal to output application specific information. A part of the application information is provided using onboard LED's. All actions are triggered using the push-button events. The events have the following meaning:

- Short press: pressing a button and releasing it under 1 second.
- Long press: pressing a button and releasing after 1 second, but before 8 seconds are passed.
- Very long press: pressing a button and releasing it after 8 seconds.

### 5.23.2 Supported platforms

The Wireless Power Transfer System Receiving Unit application is supported by the following platforms:

- FRDM-KW41Z
- USB-KW41Z
- FRDM-KW36
- FRDM-KW38

### 5.23.3 User interface

After flashing the boards, the PRU device enters a "Null" state, and the PTU enters a "Power Save" state. A short press on the ADVSW button of the PTU device simulates a device configuration and the presence of a PRU device which requires charging. For this reason, the PTU device starts a scanning procedure. A short press on the ADVSW of the PRU device simulates sufficient power to start the advertising procedure. During both scanning and advertising procedures, CONNLED will flash. If the PRU device does not connect to the PTU for 10 seconds, the advertising is stopped and CONNLED turns off. In this case the PRU device returns to the "Null" state. After the PTU connects to the PRU, CONNLED is turned on.

After the devices connect, the PTU performs device registration and determines if it has sufficient power to charge the PRU. If registration is successful, the push-buttons events change their meaning as follows:

- Short press on the PTU device triggers another scanning procedure in case another device is supported.
- Short press on the PRU device prints the dynamic parameters in the console.
- Long press on the PTU device prints information regarding the connected devices and maximum power remaining.
- Long press on the PRU device simulates a charge complete event. After the PRU is considered charged, the PTU will disconnect from it. After disconnection, the PRU device enters in a "Boot" state and requires a very long press event to return to "Null" state.
- Very long press on the PTU device simulates an error case and disconnects all connected PRU devices.
- Very long press on the PRU device simulates power removal (taking a device outside the charge area).

If the PRU device is allowed to charge, it simulates charging by printing information in the console and by turning CHGLED on.

See details below for hardware references.

Table 22. Hardware references

Platform	ADVSW	CONNLED	CHGLED
FRDM-KW41Z	SW4	LED3	LED4
FRDM-KW36	SW2	LED3	LED4-RED
FRDM-KW38	SW2	LED3	LED4-RED

### 5.23.4 Usage

The setup requires two or three supported platforms, one for PTU device and one or two for PRU devices.

1. Open serial port terminals for all the platforms. The start screen is displayed after the board is reset. At first the LEDs are flashing on all devices.
2. Press ADVSW on the PTU device and on a PRU device. CONNLED should start flashing on both devices until the connection is established or until the PRU device times out.
3. After the registration procedure is finished the PTU device will display a success message and will simulate the procedure for providing power to the PRU.

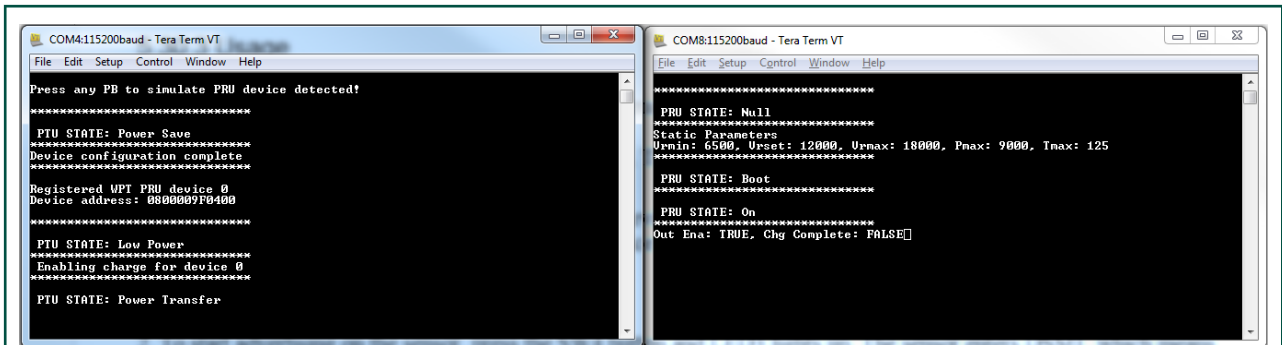


Figure 46. PTU and PRU after connecting, registering and enabling charge

4. Short press ADVSW on the PRU device again to print most recent dynamic parameters.
5. Long press on ADVSW on the PTU device to get system information.

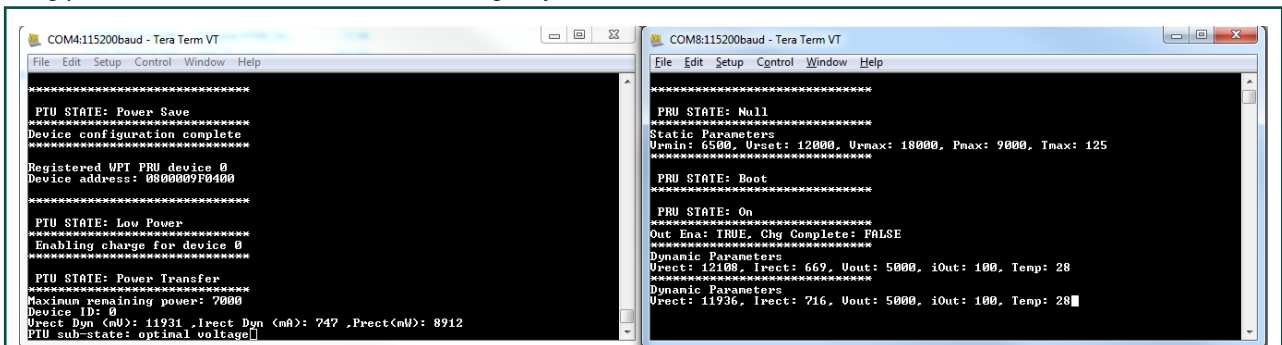


Figure 47. PTU information and PRU dynamic parameters display

6. Long press ADVSW on the PRU device to simulate a charge complete event.

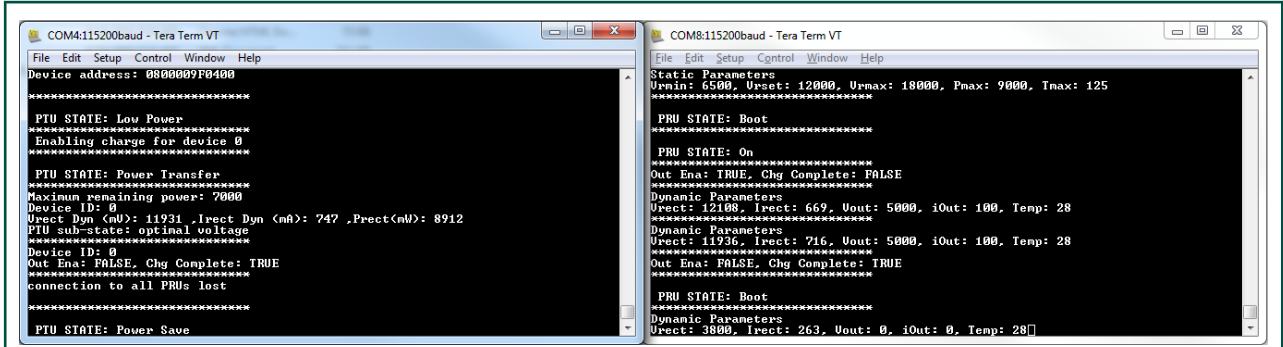


Figure 48. PRU device charged

7. Very long press ADVSW on the PRU device to simulate power removed and return it into a “Null” state.

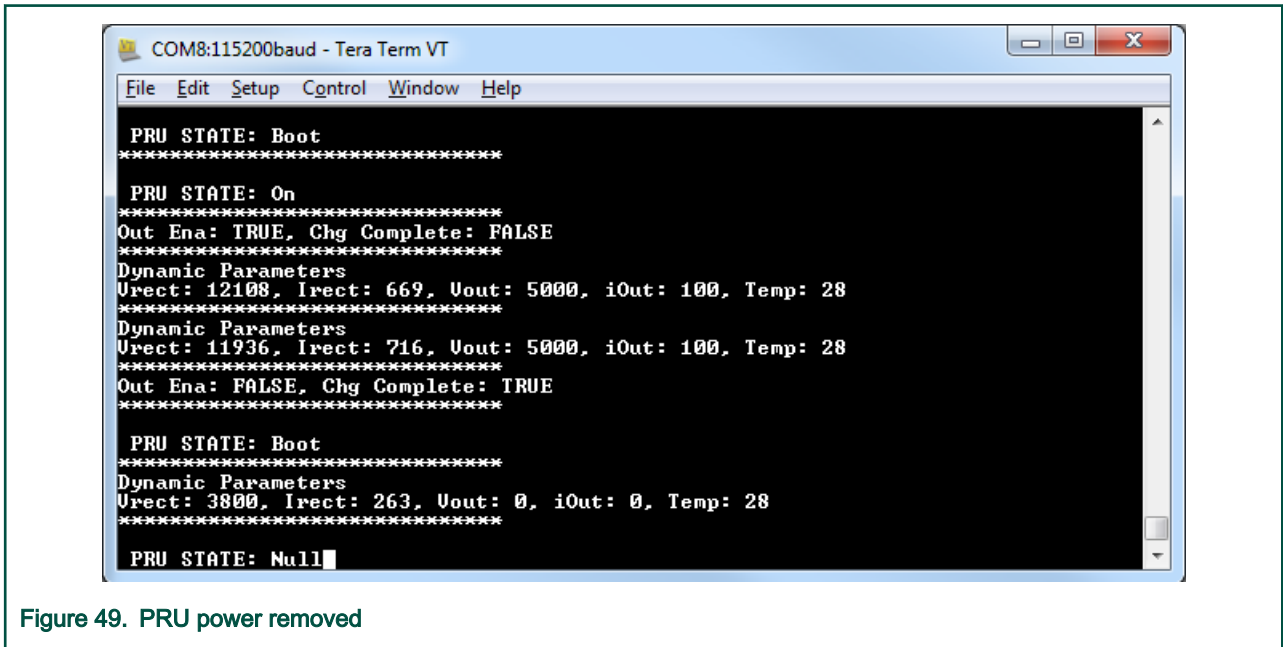


Figure 49. PRU power removed

8. If a second PRU device is used, ADVSW can be short pressed on both the PTU and on the new PRU after step 3 was finished for the previous PRU. Also, to test the power sharing procedure support, the `gAdjustPowerCommandSupport_d` macro should be defined as TRUE on at least one of the PRU devices before flashing the firmware. If the above macro is set to TRUE a scenario is designed so that two PRU devices combined require more power than the PTU device can provide, but the PTU device attempts a power sharing procedure so that it can adjust the power draw for at least one of the PRU's.

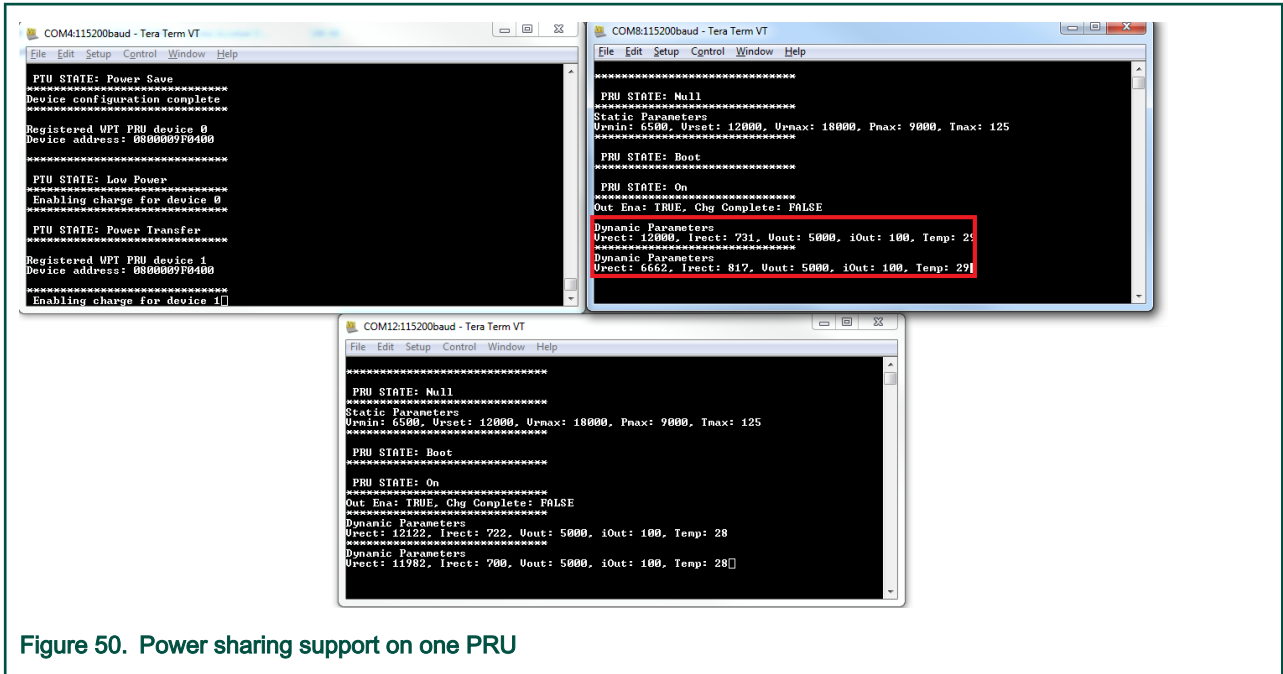


Figure 50. Power sharing support on one PRU

## 5.24 Hybrid (Dual-Mode) Bluetooth Low Energy and Generic FSK Demo Application

### 5.24.1 Implementation

The Hybrid Bluetooth LE-GFSK application demonstrates Generic FSK transmission/reception and Bluetooth advertising/scanning/multiple connections coexistence.

The Bluetooth part of this demo implements a modified version of the Wireless UART demo application.

The GenFSK part can receive/transmit data in the idle periods of the Bluetooth protocol. A predefined GFSK data packet is sent on the default channel (`gGenFskDefaultChannel_c`) at an interval (`gGenFskApp_TxInterval_c`). If at that time, the radio is in use by the Bluetooth protocol, the transmission will be postponed until the idle signal is received from the MWS coexistence module.

When in scanning mode, all received GenFSK packets will be displayed on the console along with all received Bluetooth LE events/data, preceded by a string which informs the protocol on which it was received: "BLE:" / "GFSK:".

A device can start Bluetooth advertising, scanning in parallel with GenFSK reception and transmissions. All these operations will not overlap.

Also, multiple Bluetooth connections are supported in both roles (Central and Peripheral) along with all the above.

Table 23. Application files

File name	Description
app_preinclude.h main.c	Files common to both GenFSK and Bluetooth application
app_config.c gatt_db.h gatt_uuid128.h wireless_uart.c	Bluetooth application files

Table continues on the next page...

**Table 23. Application files (continued)**

File name	Description
wireless_uart.h	
genfsk_adv.c genfsk_adv.h	GenericFSK application files

**5.24.1.1 How it works**

When a switch is pressed for the first time, the specific protocol application starts. The first protocol to start has complete access to the radio (XCVR).

When the other switch is pressed, the second protocol stack tries to start by acquiring access to the XCVR. If the first protocol is idle, access will be granted. Else it will be informed when the XCVR is idle, and the acquire control.

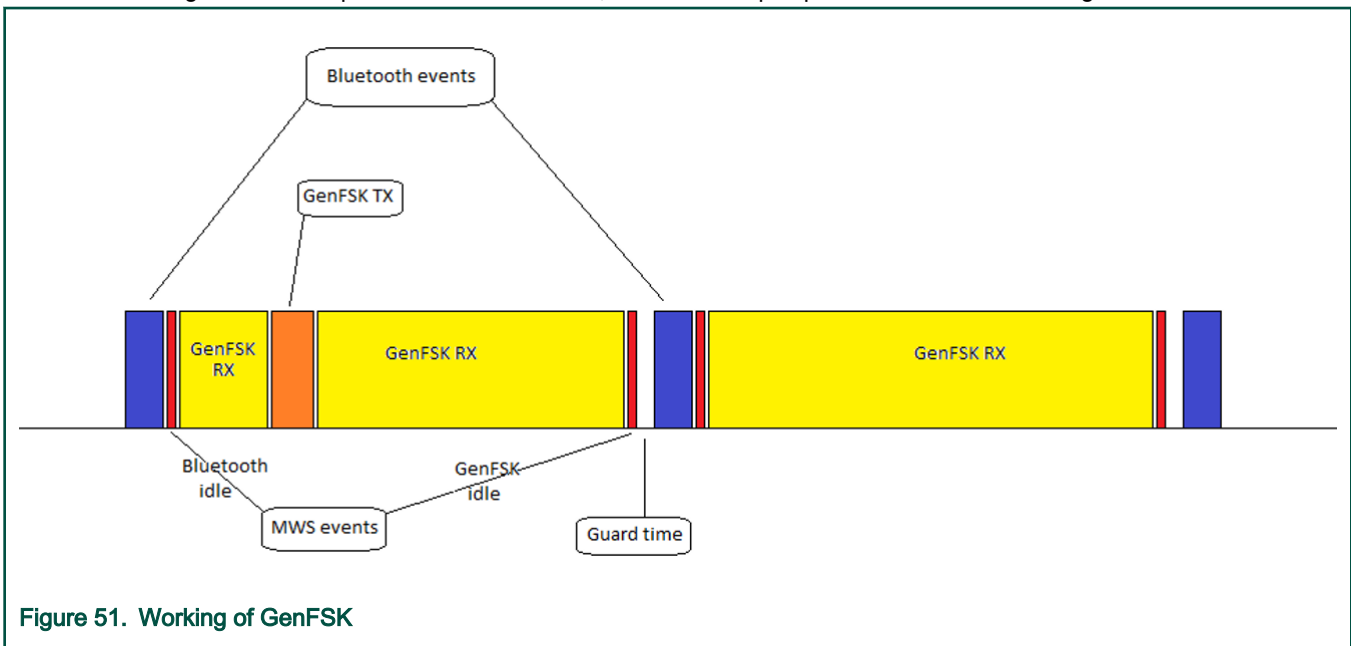
Since the Bluetooth protocol has strict timings, the GenFSK should not acquire the XCVR for long periods. Instead it should check the inactivity duration of the Bluetooth stack using the *MWS\_GetInactivityDuration()* API and release access to the XCVR before the next Bluetooth event.

Also, to maximize the GenFSK application activity, the MWS idle signal should be monitored. The demo application is doing this by registering its own callback into the MWS module using: *MWS\_GetRegisteredCallback()* API.

The GenFSK link layer also needs to be informed of the MWS events, so to achieve this, the application will first get the registered callback for the GenFSK protocol using *MWS\_GetRegisteredCallback()* API, save this callback and call it in the registered application callback. Using this method, multiple layers can chain callback to monitor MWS events.

By monitoring the MWS idle signal, the GenFSK application can start the RX operation asap. Also, to give the control to the Bluetooth stack in time, the GenFSK application starts a timer with an expire time less than the inactivity duration of Bluetooth to release access to the XCVR. Upon release, the Bluetooth stack will receive an MWS idle signal, and the handler will configure the XCVR for Bluetooth mode.

The GenFSK TX is done periodically, using a timer. If when the timer expires, the Bluetooth stack has access to the XCVR, or there is not enough idle time to perform the GenFSK TX, then it will be postponed until the next idle signal is received.



**Figure 51. Working of GenFSK**

### 5.24.2 Supported platforms

The Hybrid (Dual-Mode) Bluetooth Low Energy and Generic FSK Advertising application is supported by the following platforms:

- FRDM-KW36
- FRDM-KW38

### 5.24.3 User interface

By default, after power up, the application is in idle mode: all LEDs are flashing.

To control the application, the on-board switches are used:

- BLESW – Short press this button to start the Bluetooth advertising.
- BLESW – Long press this button to start the Bluetooth scanning.
- GFSKSW – Short press this button to start the Generic FSK transmission.
- GFSKSW – Long press this button to start the Generic FSK reception.
- CONNLED – If only this LED is flashing, the application is in advertising/scanning mode (Bluetooth LE). When the LED turns solid a Bluetooth connection has been established.

Also, a serial console application can be used to track the messages displayed, and the received Bluetooth LE/GFSK packets.

**Table 24. Hardware references**

Platform	BLESW	GFSKSW	CONNLED
FRDM-KW36	SW2	SW3	LED3
FRDM-KW38	SW2	SW3	LED3

### 5.24.4 Customization

The steps below will help changing the default settings of this demo.

For Bluetooth LE, the default advertising config (gAppAdvParams), and data (gAppAdvertisingData) are found in the app\_config.c file. Also, the scanning parameters (gScanParams) can be found in this file. The default scan interval is 100ms with a scan window of 50ms.

#### NOTE

The GFSK protocol is active during the inactive periods of the Bluetooth LE protocol. Configuring a large scan window in comparison to the scan interval will not leave enough room for the GFSK protocol to work.

For GFSK, the following defines (self-explanatory) are used by the application, and can be found in the “ble\_genfsk\_advertising.h” file:

- gGenFSK\_AdvAddress\_c
- gGenFskDefaultSyncAddress\_c
- gGenFskDefaultTxPowerLevel\_c
- gGenFskDefaultChannel\_c
- gGenFskApp\_TxInterval\_c

The “gGENFSK\_MwsControl\_c” define controls where the coexistence management should be done:



Value	Description
gGENFSK_NoMwsControl_c	No protocol coexistence management
gGENFSK_LLMwsControl_c	The GenFSK link layer manage the coexistence with Bluetooth protocol. In this case, the GenFSK will release control to the XCVR for Bluetooth after every sequence (TX/RX)
gGENFSK_AppMwsControl_c	The Application controls when the GenFSK releases the access to the XCVR. The GenFSK link layer still need to perform the XCVR configuration when the idle signal is received from the Bluetooth stack.

The payload to be transmitted over GFSK is represented by the “message” array, used by the “GfskApp\_Tx()” function.

Other GFSK settings can also be changed, but require a more advanced knowledge of the system:

- pktConfig – configures the preamble size, length field config, synch address size
- crcConfig – enable/disable the data CRC, and the polynomial used
- whitenConfig – enable/disable data whitening, polynomial used, start/end markers
- radioConfig – change data rate and modulation

### 5.24.5 Wireless usage with extended advertising

To use extended advertising with the hybrid application, make sure that the gAppUseExtendedAdvertising define is set to 1.

For FRDM-KW38, very long press BLESW button (about 10s) starts extended advertising on the coded phy. A long press BLESW button starts the scanning procedure. If a scan timeout occurs, then the application tries scanning again, but it switches the phys. If it was scanning on the 1 M phy, it starts scanning on the coded phy and vice versa. If an advertising report containing the wireless uart service is received, then a connection is initiated on the phy on which the device was found advertising (1 M for legacy advertising and coded phy for extended advertising).

To initiate a Bluetooth LE connection on coded phy, the following steps should be followed:

- Flash two boards with the ble\_w\_uart\_gfsk application and connect to each of them using a serial port.
- Very long press the BLESW button of the board you want to act as the peripheral.

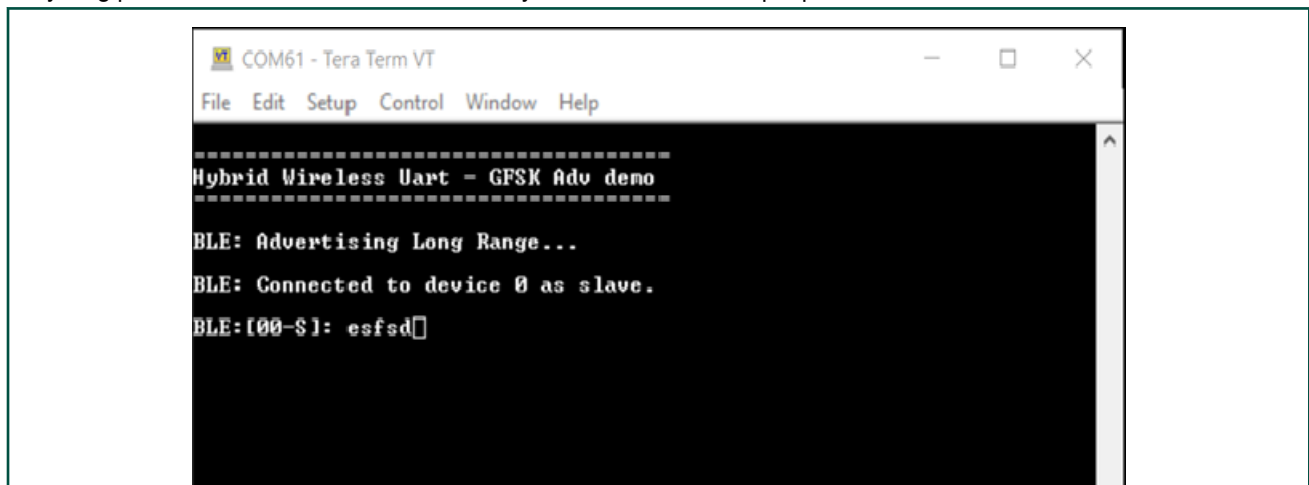


Figure 52. Long Range advertising connection slave

- Long press the BLESW button of the board you want to act as the central.

- Once the connection is established, you can send characters between the master and slave.

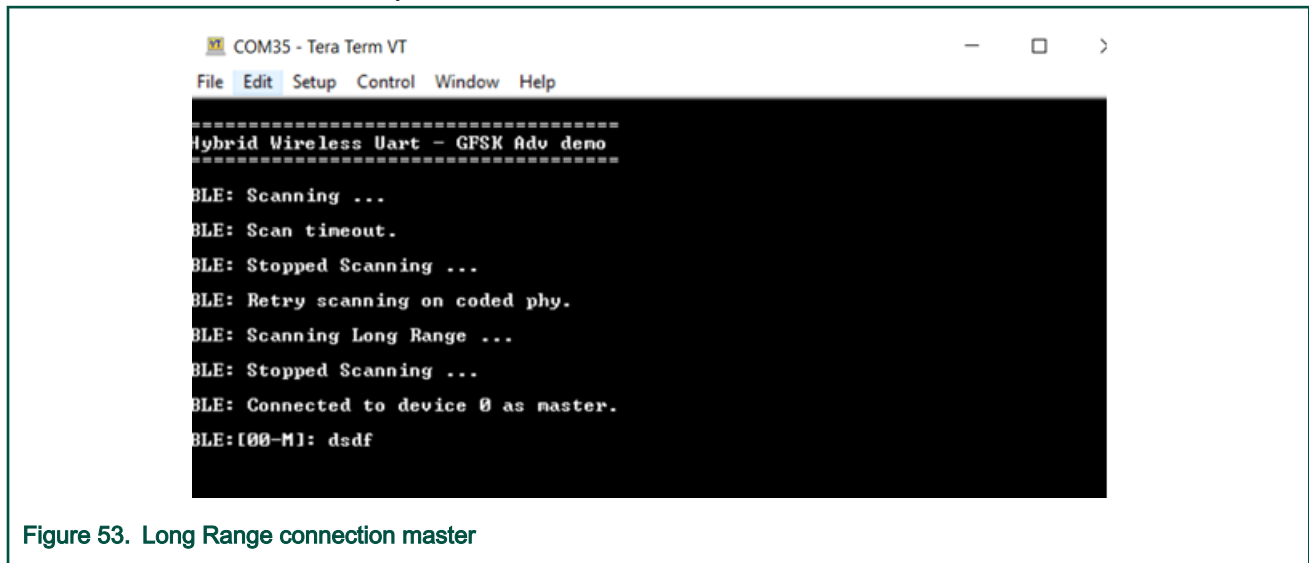


Figure 53. Long Range connection master

#### NOTE

The GFSK Tx/Rx behavior is still available when using extended advertising.

## 5.25 ANCS Client

This section describes the implemented profiles, services, user interactions, and testing methods for the Bluetooth LE OTAP application.

### 5.25.1 Implemented profile and services

The ANCS Client application implements an ANCS Client for the custom ANCS Service available on iOS mobile devices.

Check the documentation available on the iOS website for details about the ANCS service, its characteristics and supported features.

The demo application acts as a GAP Peripheral which advertises a service solicitation for the custom ANCS Service. It also acts as a GATT Client once connected to a device offering the ANCS Service. The application offers some services as GATT Server.

Once connected to a mobile device offering the ANCS Service, the application displays information about ANCS Notifications received from that device. The notifications are received via ATT Notifications for which the ANCS Client must register on the peer ANCS Server. The application also retrieves and displays additional information about the received ANCS Notifications. This is achieved by writing commands to specified characteristics on the ANCS Server and receiving responses via ATT Notifications from other characteristics. All information is displayed to the user using a shell available over a serial communications interface.

Accessing the ANCS Service requires Bluetooth LE security to be enabled.

### 5.25.2 Supported platforms

The ANCS Client application is supported on the following platforms:

- FRDM-KW36
- FRDM-KW38

### 5.25.3 User interface

After flashing the board, the device is in idle mode (all LEDs flashing). To start advertising, press the ADVSW button. When in GAP Discoverable Mode, CONNLED is flashing. When the ANCS Server (Gap Central) connects to the ANCS Client (GAP

Peripheral), CONNLED turns solid. To disconnect, hold the ADVSW for 2-3 seconds. The ANCS Client then re-enters the advertising state.

For displaying operating information and ANCS Notifications, the demo application uses a shell exposed via a serial communication interface.

See details below for hardware references.

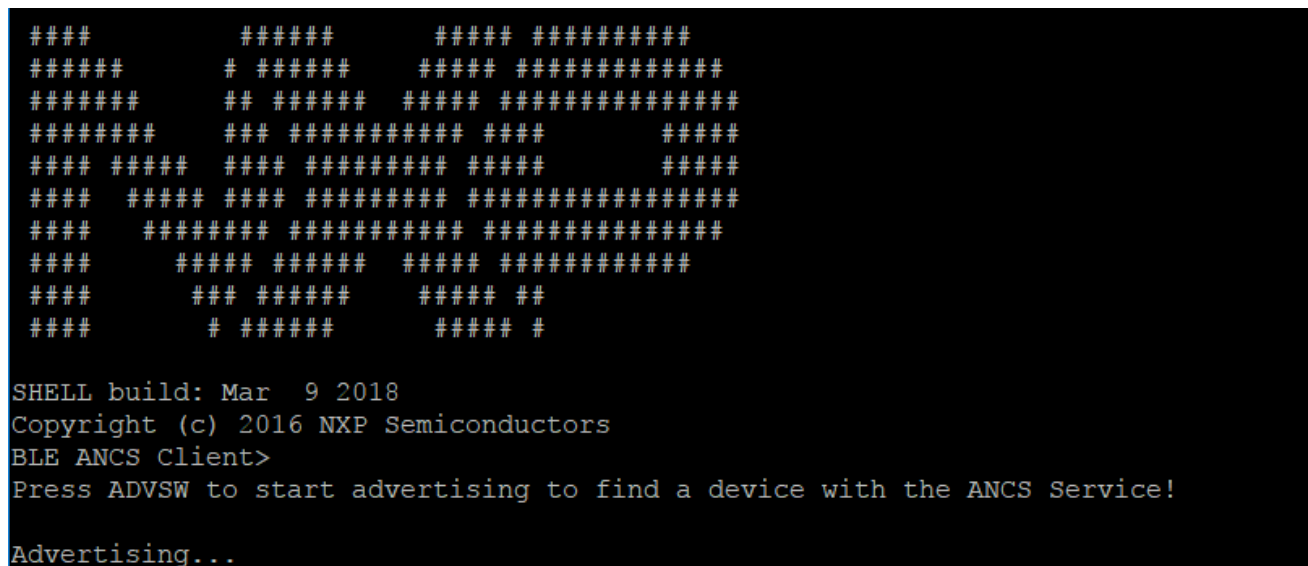
**Table 25. Hardware references**

Platform	ADVSW	CONNLED
FRDM-KW36	SW2	LED3
FRDM-KW38	SW2	LED3

### 5.25.4 Usage

The ANCS Client demo application is designed to work with a peer mobile device which exposes the ANCS service. Also, a serial terminal application is required for displaying ANCS Notifications information.

- Open a serial terminal application on the PC and connect it to the serial port corresponding to the board on which the ANCS Client runs. See the details in Section 5.1.3, "User Interface". A start screen is displayed immediately after the board is reset. All LEDs should be flashing.



- Press the ADVSW to start advertising. This instruction is also displayed in the serial terminal as shown in the screenshot above.
- The peer device starts scanning for Bluetooth LE devices and connect to the ANCS Client device which is advertising.
- Once connected to a peer the application, it looks for the ANCS Service and its characteristics. If they are found, the ANCS Client tries to register for receiving notifications.
- If any security related ATT errors are encountered than the application automatically performs Pairing and Bonding and retries the failed ATT operations. Depending on the negotiated Pairing Method, user interaction may be needed to complete the Pairing. Follow the on-screen instructions provided by both the ANCS Client and the mobile device. If a

passkey is to be generated by the ANCS Client then the default 999999 passkey is used.

```

Pairing was successful! Pairing completed successfully

Writing ANCS Notification Source CCCD... Enabling ANCS Notifications

ANCS Notification Source CCCD written successfully. Success!

Writing ANCS Data Source CCCD...

ANCS Data Source CCCD written successfully. Success!
    
```

- After bidirectional communication is established via GATT the ANCS Client will start displaying ANCS Notifications information as shown below.

```

BLE ANCS Client>
Notif_UID  Flags  Category  Application  3 existing notifications, 1 new
0x00000000  S_E_N  Social    Hangouts    notification for which the
0x00000001  S__N   Social    Hangouts    application name is not
0x00000002  S__N   Social    Hangouts    available yet
0x00000003  ___N   Email

BLE ANCS Client>
Notif_UID  Flags  Category  Application  The application name for the
0x00000000  S_E_N  Social    Hangouts    new notification has been
0x00000001  S__N   Social    Hangouts    obtained from the ANCS Server
0x00000002  S__N   Social    Hangouts
0x00000003  ___N   Email    Mail

BLE ANCS Client>
Notif_UID  Flags  Category  Application  One notification was deleted
0x00000000  S_E_N  Social    Hangouts    by the ANCS Server and it has
0x00000001  S__N   Social    Hangouts    notified the ANCS Client to
0x00000002  S__N   Social    Hangouts    delete it too.

BLE ANCS Client>
Notif_UID  Flags  Category  Application  A second notification is deleted
0x00000001  S__N   Social    Hangouts    by the ANCS Server.
0x00000002  S__N   Social    Hangouts

BLE ANCS Client>
Notif_UID  Flags  Category  Application  A third notification is deleted
0x00000002  S__N   Social    Hangouts    by the ANCS Server.

BLE ANCS Client>
Notif_UID  Flags  Category  Application  The last notification is deleted
0x00000002  S__N   Social    Hangouts    by the ANCS Server.

BLE ANCS Client>
Notif_UID  Flags  Category  Application  1 new notification is received
0x00000004  ___N   Social    Hangouts    from the ANCS Server for which
the application name is not
available yet.

BLE ANCS Client>
Notif_UID  Flags  Category  Application  The application name for the
0x00000004  ___N   Social    Hangouts    last notification is retrieved from
the ANCS Server.

BLE ANCS Client>
    
```

## 5.26 Bluetooth LE FSCI Black Box

This section describes the functionality, user interactions, and testing methods for the Bluetooth LE FSCI Black Box demo application.

### 5.26.1 Description

The Bluetooth LE FSCI Black Box demo application gives access to the Bluetooth LE Host Stack via a serial interface using the FSCI protocol. See the *FSCI (Framework Serial Communication Interface) manual* for the format of the FSCI commands and a full list of supported commands.

The demo can be used with the Test Tool for Connectivity Products. Command Console application can be downloaded from the NXP website or with a custom application which supports the FSCI protocol and commands.

### 5.26.2 Supported platforms

The Bluetooth LE FSCI Black Box application is supported on the following platforms:

- FRDM-KW41Z
- USB-KW41Z
- QN9080XCDK
- FRDM-KW36
- FRDM-KW38
- USB-KW38

### 5.26.3 Usage with Test Tool for Connectivity Products

The Bluetooth LE FSCI Black Box demo application is designed to be used via serial interface. This can be done using the TEST Tool for Connectivity Products – Command Console application as described below.

1. Download the demo application onto a supported board
2. Connect the board to a USB port of the PC. The UASB COM port drivers must be installed properly and a COM port corresponding to the board should be available.
3. Open the Test Tool application and connect to the serial port corresponding to the board on which the Bluetooth LE FSCI Black Box application runs. The serial communication parameters are: baud rate 115200, 8N1

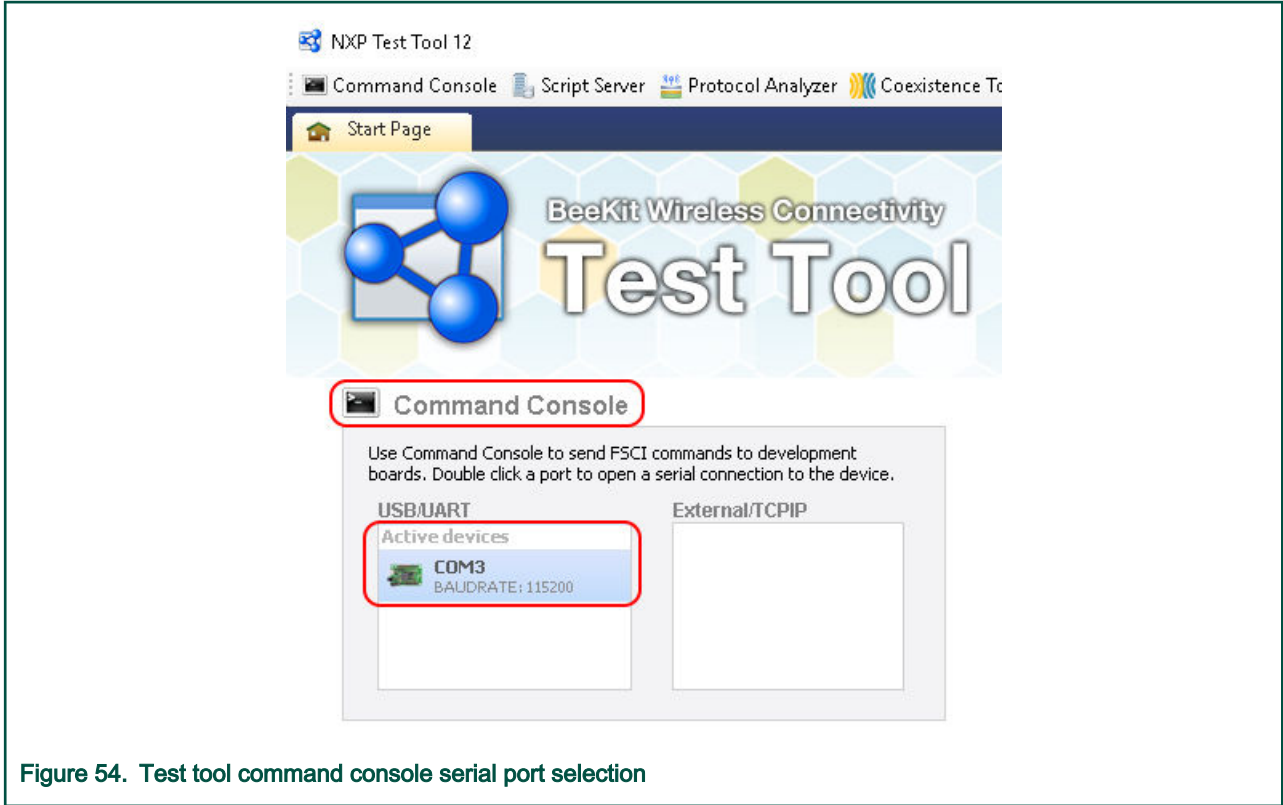


Figure 54. Test tool command console serial port selection

4. Select the appropriate Test Tool XML file from the drop-down list for the release being used and send some commands to the application.

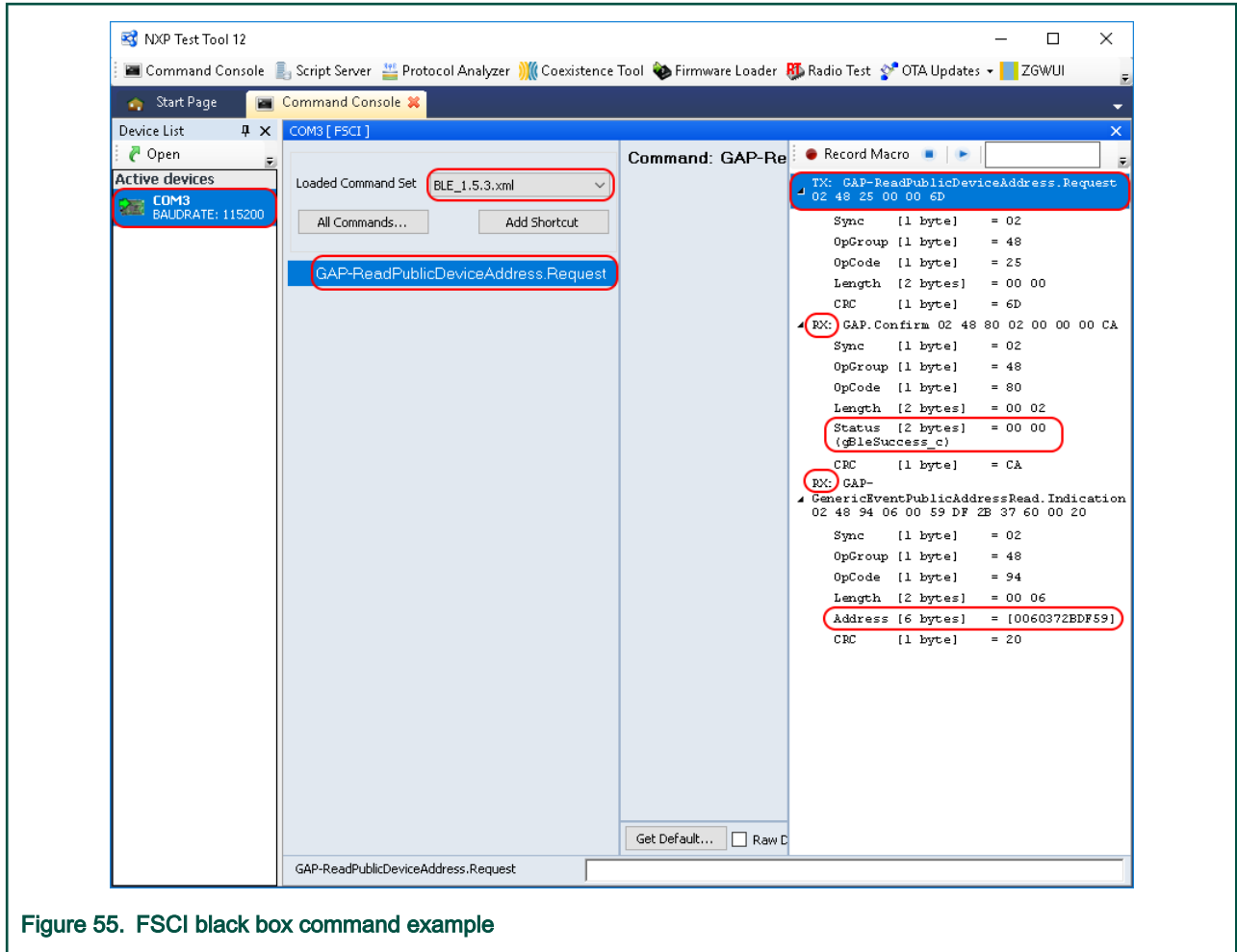


Figure 55. FSCI black box command example

## 5.27 HCI Black Box

This section describes the functionality, user interactions, and testing methods for the Bluetooth LE HCI Black Box demo application.

### 5.27.1 Description

The Bluetooth LE HCI Black Box demo application gives access to the Bluetooth LE Controller via a serial interface using the HCI protocol over serial interface. See the Bluetooth Specification for the format of HCI commands and events over serial interface and the full list of supported commands and events.

The demo can be used with the Test Tool for Connectivity Products. Command Console application can be downloaded from the NXP website or with a custom application which supports the HCI protocol and commands and events over serial interface.

Note that the HCI protocol encapsulation is dependent on the type of interface it is being used on. See the Bluetooth Specification for the HCI message format on each type of supported interface.

For instructions using the Bluetooth LE HCI Black Box with a serial terminal application or the *hcitool* in Linux, check this [article](#)

### 5.27.2 Supported platforms

The HCI Black Box application is supported on the following platforms:

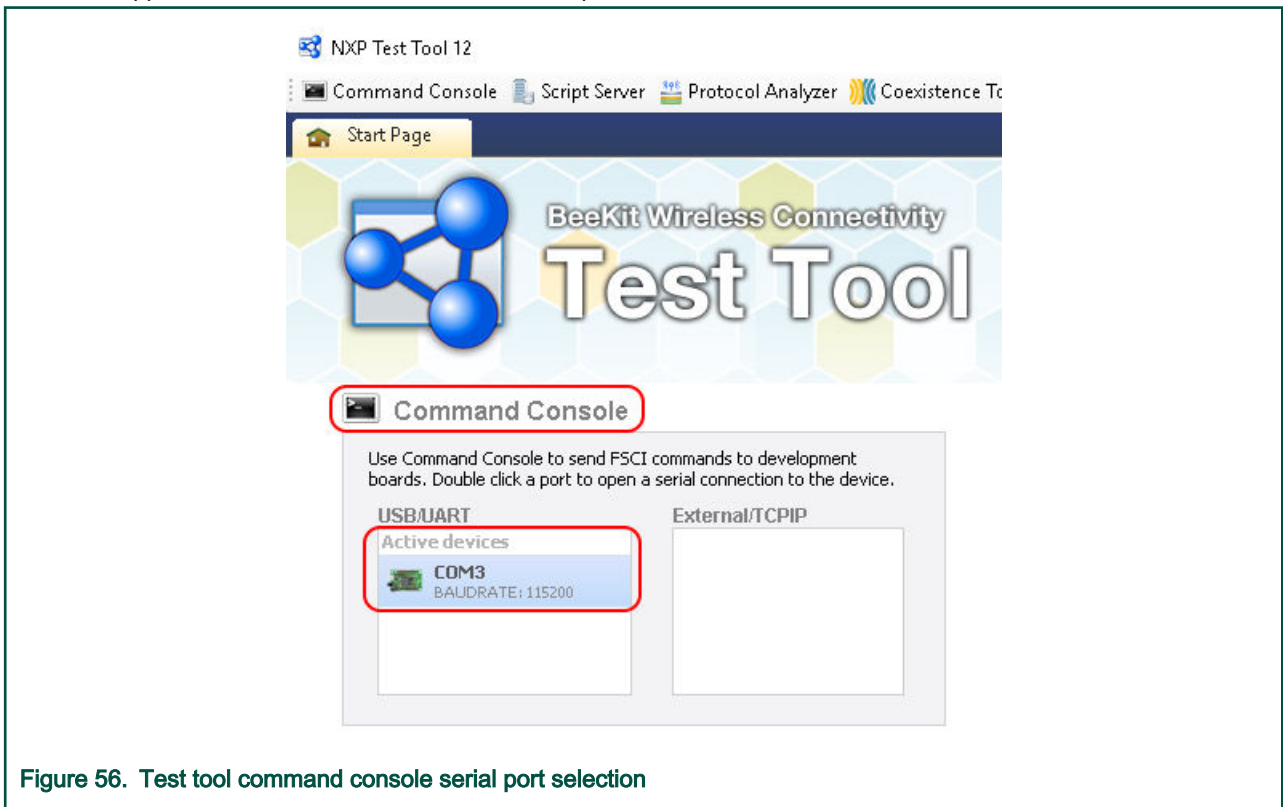
- FRDM-KW41Z
- USB-KW41Z

- QN9080XCDK
- FRDM-KW36
- FRDM-KW38
- USB-KW38

### 5.27.3 Usage with Test Tool for Connectivity Products

The Bluetooth LE HCI Black Box demo application is designed to be used via serial interface. This can be achieved using the TEST Tool for Connectivity Products – Command Console application as described below.

1. Download the demo application to a supported board
2. Connect the board to an USB port of the PC. The UASB COM port drivers must be installed properly and a COM port corresponding to the board should be available.
3. Open the Test Tool application and connect to the serial port corresponding to the board on which the Bluetooth LE HCI Black Box application runs. The serial communication parameters are: baud rate 115200, 8N1



**Figure 56. Test tool command console serial port selection**

4. Select the appropriate Test Tool HCI XML file from the drop-down list for the release you are using and send some commands to the application.



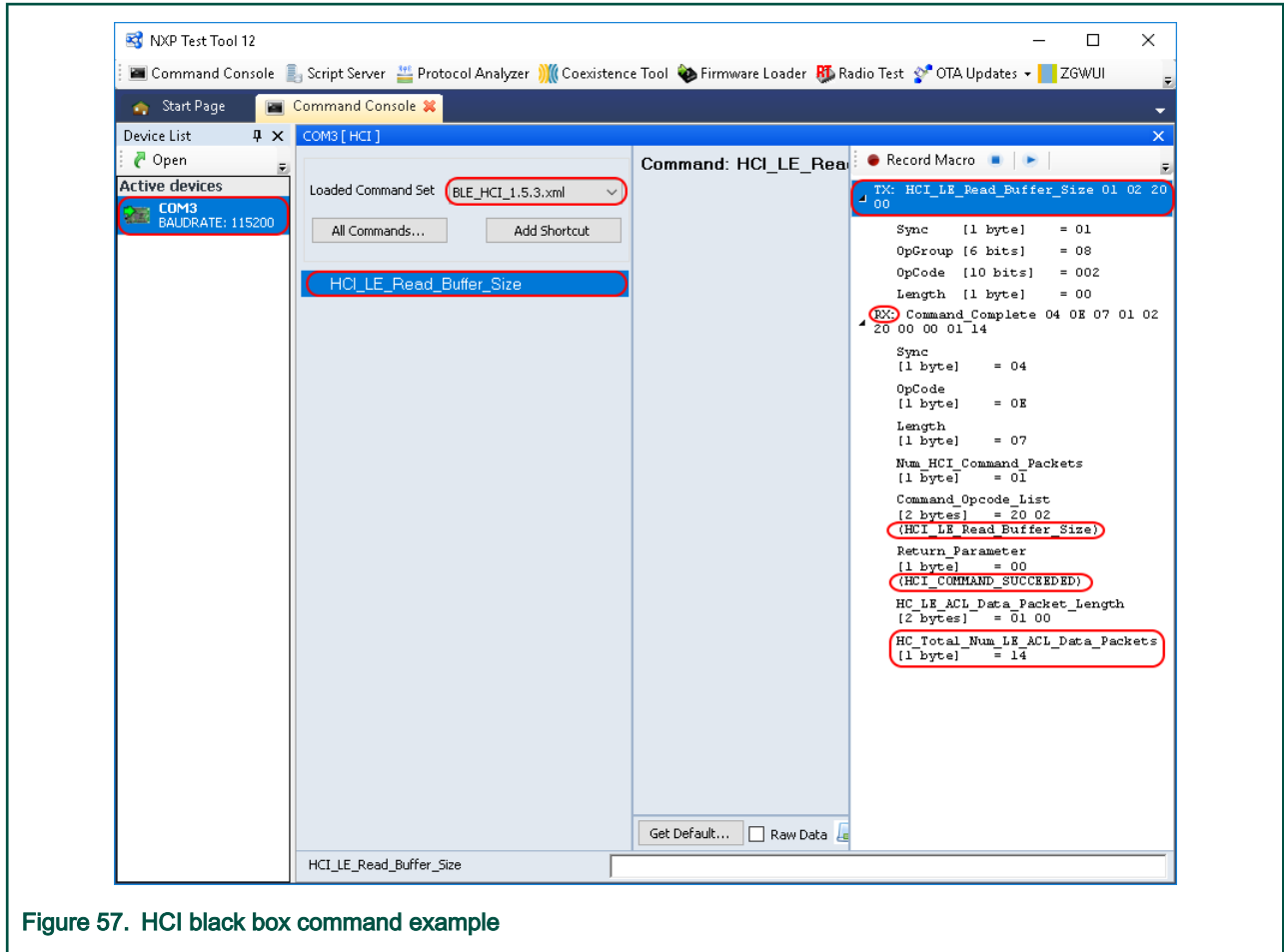


Figure 57. HCI black box command example

### 5.27.4 Direct Test Mode

HCI Black Box application can be used to test radio functionality at physical layer through Direct Test Mode (DTM) described in *Bluetooth Specification Version 5.0, Vol 6, Part F*. The testing done so can cover physical layer characteristics such as:

- Transmission power and receiver sensitivity
- Frequency offset and drift
- Modulation characteristics
- Packet error rate
- Intermodulation performance

DTM supports two interfaces for communication:

- Over HCI (enabled by default in HCI Black Box)
- Over 2-Wire UART interface

To use DTM over 2-Wire UART, it must be enabled setting *gEnabledDTM\_d* on TRUE and connecting to the proper pins on the board. The default baud rate speed is set to 115200 but it can be changed by configuring *gDefaultDTMBaudrate(For 2-Wire UART)* or *gHciInterfaceSpeed\_d(For HCI)*;

For DTM pin connection, refer to Pinout chapter in Reference Manual.

## 5.28 Pulse Oximeter Sensor

This section describes the implemented profiles and services, user interactions, and testing methods for the Pulse Oximeter Sensor application.

### 5.28.1 Implemented profile and services

The Pulse Oximeter Sensor application implements a GATT server and the following profiles and services.

- Pulse Oximeter Service v1.0
- Battery Service v1.0
- Device Information Service v1.1

The application behaves as a GAP peripheral node. It enters GAP General Discoverable Mode and waits for a GAP central node to connect. When a peer device connects and configures notifications or indications, the sensor starts sending blood oxygen saturation measurements at regular intervals.

### 5.28.2 Supported platforms

The Pulse Oximeter Sensor application is supported on the following platforms:

- FRDM-KW41Z
- USB-KW41Z
- QN9080XCDK
- FRDM-KW36
- FRDM-KW38
- QN9090/K32W061

### 5.28.3 User interface

After flashing the board, the sensor is in idle mode (all LEDs flashing). To start advertising, press the ADVSW button. When in GAP Discoverable Mode, CONNLED is flashing. When a central node connects to the peripheral, CONNLED turns solid. To disconnect the node, hold the ADVSW button for 2-3 seconds. The node then re-enters GAP Discoverable Mode.

The location and navigation measurement values are generated randomly on the device.

See details below for hardware references.

**Table 26. Hardware references**

Platform	ADVSW	CONNLED
FRDM-KW41Z	SW4	LED3
QN9080CDK	BUTTON1	D5-RED
FRDM-KW36	SW2	LED3
FRDM-KW38	SW2	LED3

### 5.28.4 Usage

The application can be tested against the Profile Tuning Suite (PTS) black-box testing tool from the Bluetooth SIG.

## 5.29 Location and Navigation Server and Client

This section describes the implemented profiles and services, user interactions, and testing methods for the Location and Navigation Server and Client applications.

### 5.29.1 Implemented profile and services

The Location and Navigation Server application implements a GATT server and the following profiles and services:

- Location and Navigation Service v1.0
- Battery Service v1.0
- Device Information Service v1.1

The Location and Navigation Server application behaves as a GAP peripheral node. It enters GAP General Discoverable Mode and waits for a GAP central node to connect. The Location and Navigation Client application behaves as a GAP central node. It scans for a GAP peripheral node to connect to. Once the peers are connected and notifications and indications are configured, the server starts sending location and navigation measurements at regular intervals.

### 5.29.2 Supported platforms

The Location and Navigation Server and Client applications are supported on the following platforms:

- FRDM-KW36
- FRDM-KW38

### 5.29.3 User interface

After flashing the board with the server application, the sensor is in idle mode (all LEDs flashing). To start advertising, press the ADVSW button. When in GAP Discoverable Mode, CONNLED is flashing. After flashing the board with the client application, the device is in idle mode (all LEDs flashing). To start scanning, press the SCANSW button. When scanning, CONNLED is flashing. When the client and server are connected, CONNLED turns solid on both. To disconnect the server, hold the ADVSW button for 2-3 seconds. The node then re-enters GAP discoverable Mode.

**Table 27. Hardware references**

Platform	ADVSW/SCANSW	CONNLED
FRDM-KW36	SW2	LED3
FRDM-KW38	SW2	LED3

### 5.29.4 Usage

The two applications are tested together.

- Open a serial terminal application on the PC and connect it to the serial port corresponding to the board on which the Client runs.
- Press ADVSW on the Server.
- Press SCANSW on the Client.
- Observe the notifications arriving periodically on the Client.

```
Scanning...
Found device:
NXP_LNS
00 60 37 46 DC CA
Connected!

Received Report:
=====
UTC Time: 6537216

Received Report:
=====
Instantaneous Speed: 17
Total Distance: 436207617
Location: 537250074, 2584109489
Elevation: 1403

Received Report:
=====
UTC Time: 6542368
```

Figure 58. Notifications arriving periodically

# Chapter 6

## Revision history

This table summarizes revisions to this document.

**Table 28. Revision history**

Revision number	Date	Substantive changes
0	06/2015	Initial release
1	10/2015	Added new applications
2	04/2016	Adapted the text and code extracts in OTAP chapter to match the new Bluetooth Low Energy 4.2 implementation changes.  Added section that describes how to create an OTAP image file from a BIN type file.  Added more detailed explanations and diagrams to the Bootloader section.  Added LE Long Frames section.  Updated Low Power section.  Updated RTOS section.  Added Enhanced Privacy section.  Added Dynamic GATT Database section.  Updated GAP section with LE Secure Connections references.
3	07/2016	Updated the Application Structure section.
4	09/2016	Public information
5	12/2017	Adapted the text to match the QN9080 platform which supports Bluetooth Low Energy 5.0
6	03/2018	Updated for KW36
7	06/2018	Updated for KW35
8	04/2019	Updated for KW37EAR Release
9	07/2019	Updated for KW37/38/39
10	12/2019	Updated for KW37 PRC 2.0
11	02/2020	Updated for KW37 PRC 3.0
12	04/2020	Updated for KW37 RFP
13	06/2020	Updated for KW35 Maintenance Release4

## How To Reach Us

### Home Page:

[nxp.com](http://nxp.com)

### Web Support:

[nxp.com/support](http://nxp.com/support)

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: [nxp.com/SalesTermsandConditions](http://nxp.com/SalesTermsandConditions).

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer's applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, Altivec, CodeWarrior, ColdFire, ColdFire+, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, Tower, TurboLink, EdgeScale, EdgeLock, eIQ, and Immersive3D are trademarks of NXP B.V. All other product or service names are the property of their respective owners. AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© NXP B.V. 2016-2020.

All rights reserved.

For more information, please visit: <http://www.nxp.com>

For sales office addresses, please send an email to: [salesaddresses@nxp.com](mailto:salesaddresses@nxp.com)

Date of release: 12 May 2020

Document identifier: BLEDAUG

