BLEDAUG

Bluetooth Low Energy Demo Applications User’s Guide

Rev. 3.6 — 26 November 2024 **User guide**

Document information

Information	Content
Keywords	Bluetooth Low Energy host stack, KW45, K32W1, FRDM-MCXW71, KW47, MCXW72 development platforms, evaluation kit (EVK), KW47-EVK, MCXW72-EVK and FRDM-MCXW72 evaluation kits, KW45B41Z-EVK, K32W148-EVK, , hardware and toolchain requirements, Software Development Kit (SDK), sample applications
Abstract	This document describes the Bluetooth Low Energy host stack enablement for KW45B41Z-EVK, K32W148-EVK, FRDM-MCXW71, KW47-EVK, MCXW72-EVK, and FRDM-MCXW72 development platforms development platforms. It describes the hardware and toolchain requirements and steps to build and run the demo applications included in the software development kit.

1 Introduction

This document describes the Bluetooth Low Energy host stack enablement for NXP development platforms. The document is organized as follows:

- Section 2 "Bluetooth Low Energy applications" lists the demo applications that can be located in the Software Development Kit (SDK).
- Section 3 "Hardware configurations" describes the hardware and toolchain requirements.
- Section 4 "Building and running a Bluetooth LE example application" describes the general requirements for using and testing a Bluetooth Application on a compatible device.
- Section 5 "Bluetooth LE stack and demo applications" describes the steps and instructions for using the demo applications on your device. It also presents the profiles and services implemented and how to interact with them.
- Section 6 "References" lists the additional documents that can be referred for more information.
- Section 7 "Acronyms" lists the acronyms used in this document.

2 Bluetooth Low Energy applications

The Software Development Package provides a Bluetooth Low Energy v5.3-compliant host stack implementation with a set of GATT-based profiles and services implemented on top. To demonstrate the device functionality, the following demo applications are implemented.

1. ANCS Client
2. Beacon Application
3. Bluetooth LE FSCI Black Box
4. EATT Central
5. EATT Peripheral
6. HCI Black Box
7. HID Host
8. HID Device (Mouse)
9. Low-power Temperature Sensor and Collector
10. Low-power Extended Advertising Central and Peripheral
11. OTAP Clients ATT and L2CAP and OTAP Server
12. Wireless UART demo application
13. Bluetooth LE Shell application
14. Hybrid (Dual-Mode) Bluetooth Low Energy and Generic FSK

Note: *Refer to the application notes that are located in the '`documentation`' folder.*

3 Hardware configurations

3.1 Hardware requirements

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

- KW45B41Z-EVK
- K32W148-EVK
- FRDM-MCXW71
- KW47-EVK
- MCXW72-EVK
- FRDM-MCXW72

3.2 Toolchain requirements

The Bluetooth Low Energy Stack demo applications were compiled and tested with IAR Embedded Workbench for Arm and MCUXpresso. Users must use one of these tools.

3.3 KW45B41Z-EVK platform

Figure 1 displays the top view of the KW45B41Z-EVK board.

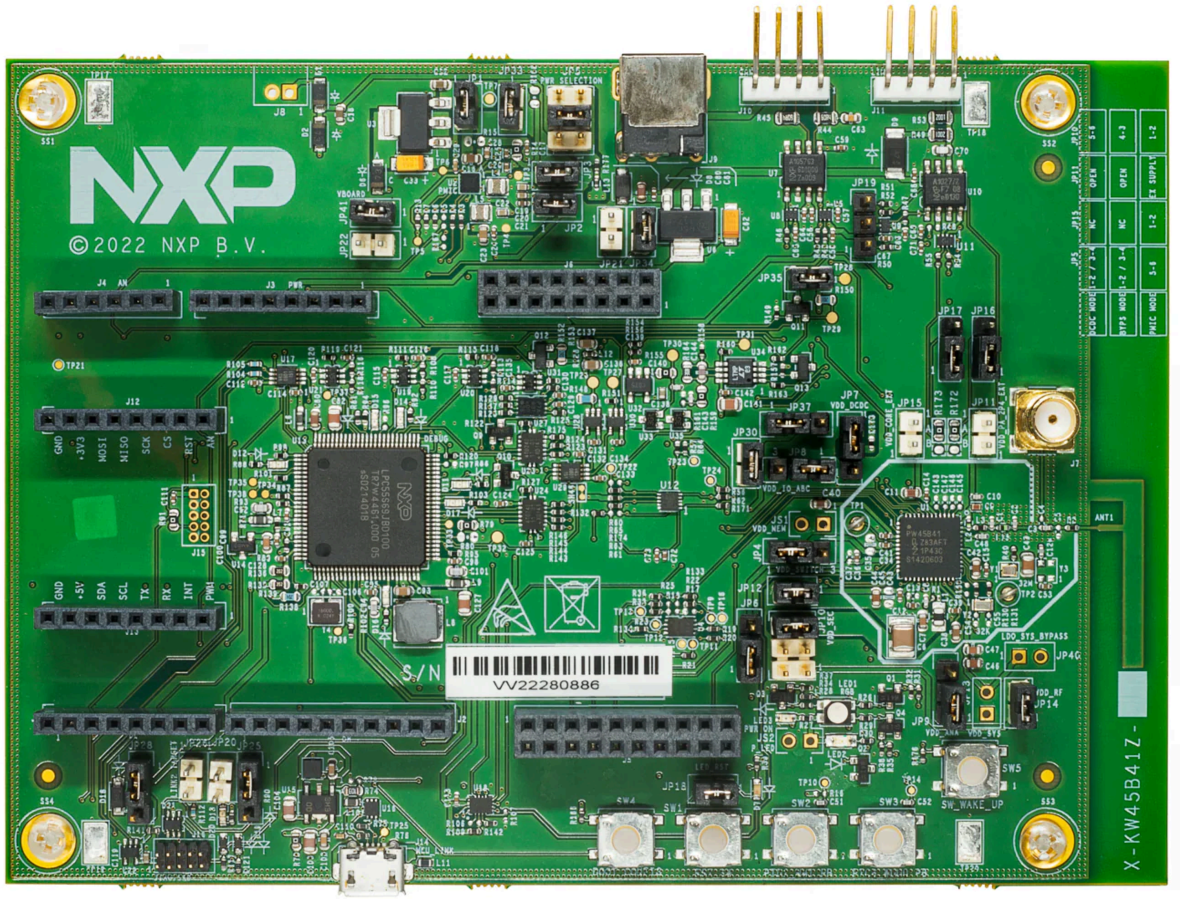


Figure 1. KW45B41Z-EVK board

3.4 K32W148-EVK platform

Figure 2 displays the top view of the K32W148-EVK board.

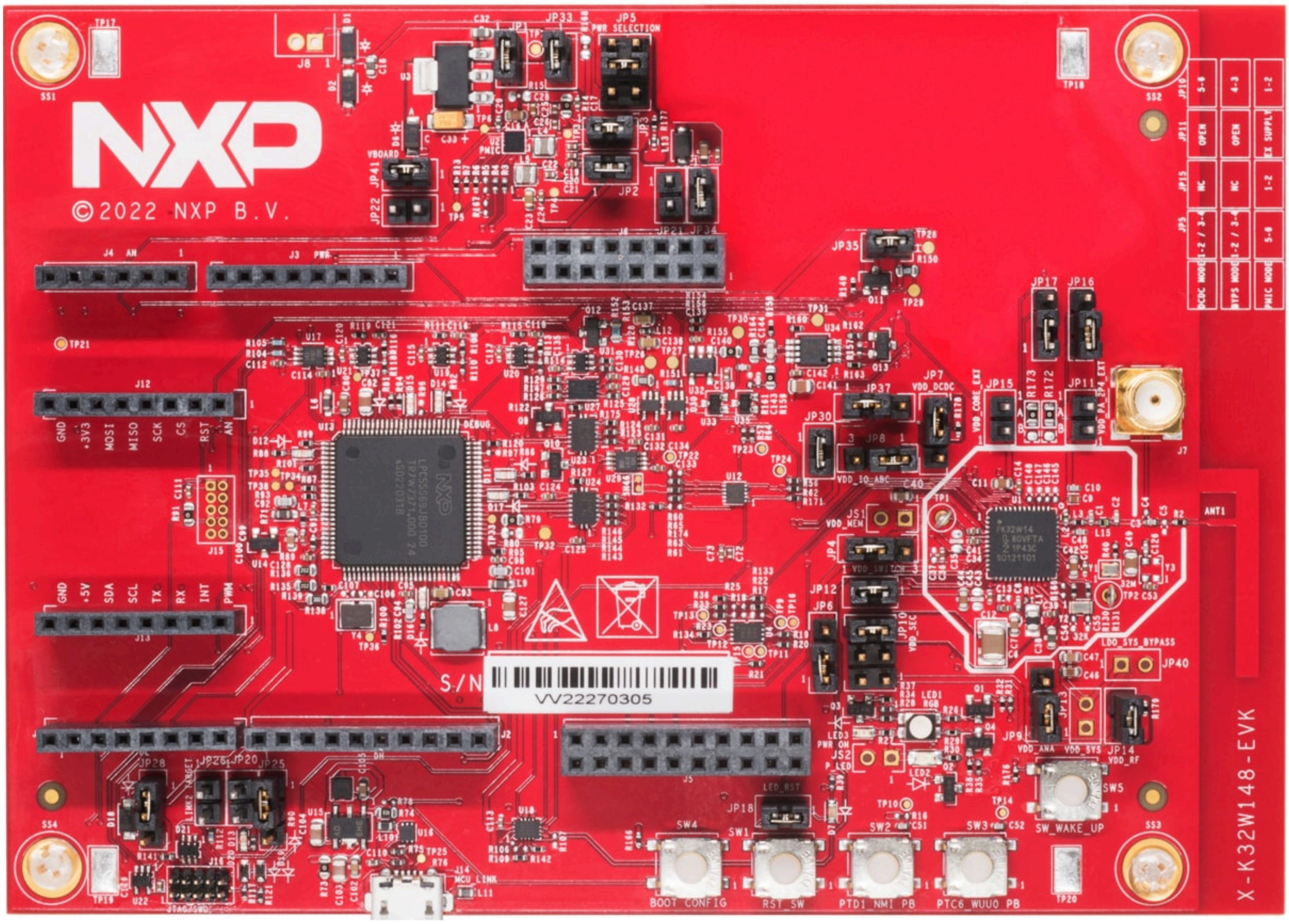


Figure 2. K32W148-EVK board

3.5 FRDM-MCXW71 platform

Figure 3 displays the top view of the FRDM-MCXW71 board.

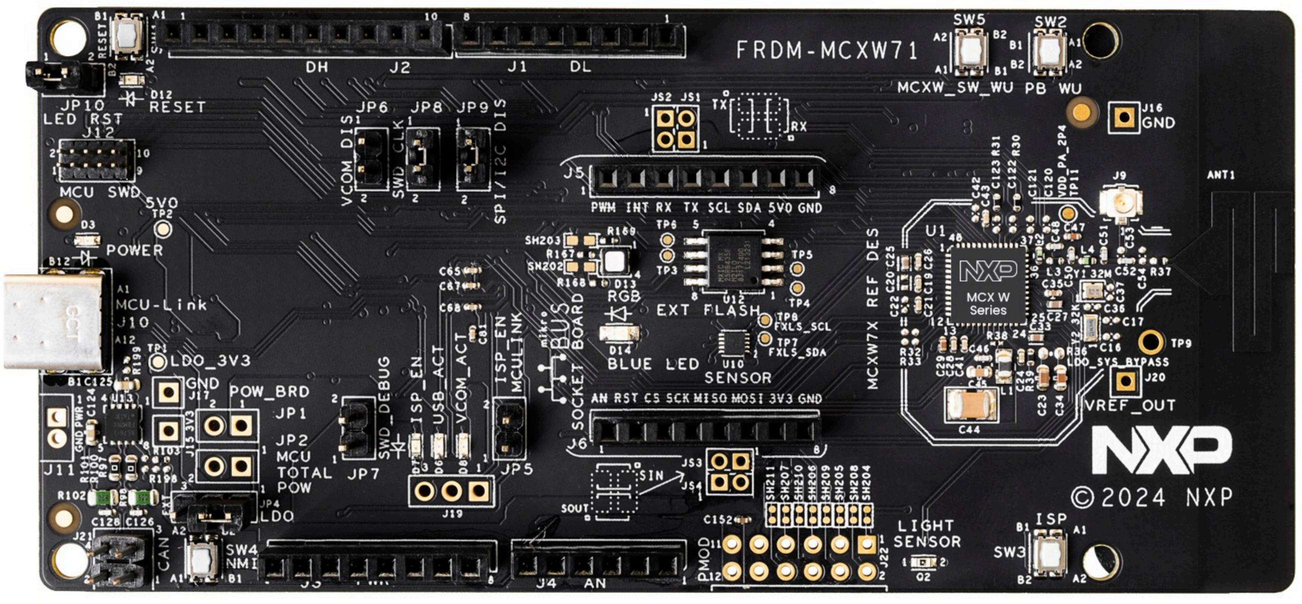


Figure 3. FRDM-MCXW71 board

4 Building and running a Bluetooth LE example application

This section presents the general requirements for using and testing a demo application. To open, build, and run any example application on a specific board, refer to the Bluetooth Low Energy Quick Start Guide document for the corresponding board.

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 display various information regarding the current software, hardware, and firmware revisions. These values are used as an example and application developers can modify them when developing their product. The values can be read from the device through a connected GATT client.

4.2 Security

The examples that enable pairing always generate a default passkey of 999999 that must be entered on the Central device, which is usually a smartphone or tablet.

4.3 Testing devices

To demonstrate the profile functionality, most of the scenarios require one of the supported platforms and a Bluetooth Low Energy capable central device. The device is usually a smartphone or a tablet that runs a compatible Bluetooth LE application. Figure 4 shows the **IoT Toolbox** UI.

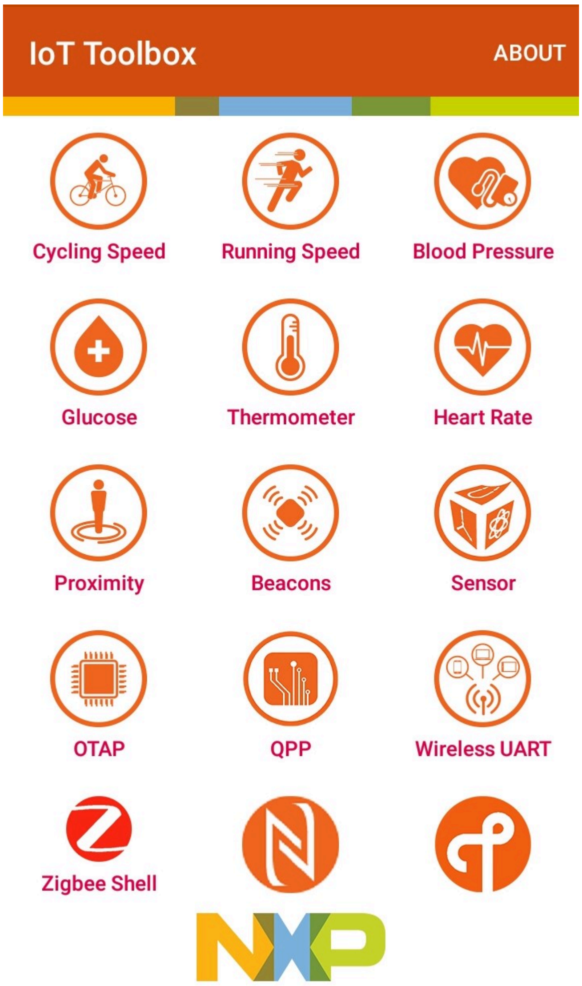


Figure 4. IoT Toolbox

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 Apple Playstore or on Google Play.

Other demos can be run by using two platforms, one for the peripheral and one for the central role. A few examples of such demos are listed below:

- Low-Power Temperature Sensor and Collector
- Wireless UART
- OTAP Client and Server
- HID Host and Device
- Extended Advertising Central and Peripheral
- EATT Central and Peripheral

To provide feedback and more interaction, some examples use a shell console via the virtual COM port. To access the device, open a serial port terminal and as shown in the Figure 5. For this example, Tera Term VT and a KW45B41Z-EVK or K32W148-EVK, or a FRDM-MCXW71 board can be used. The communication parameters are 115200 and 8N1.

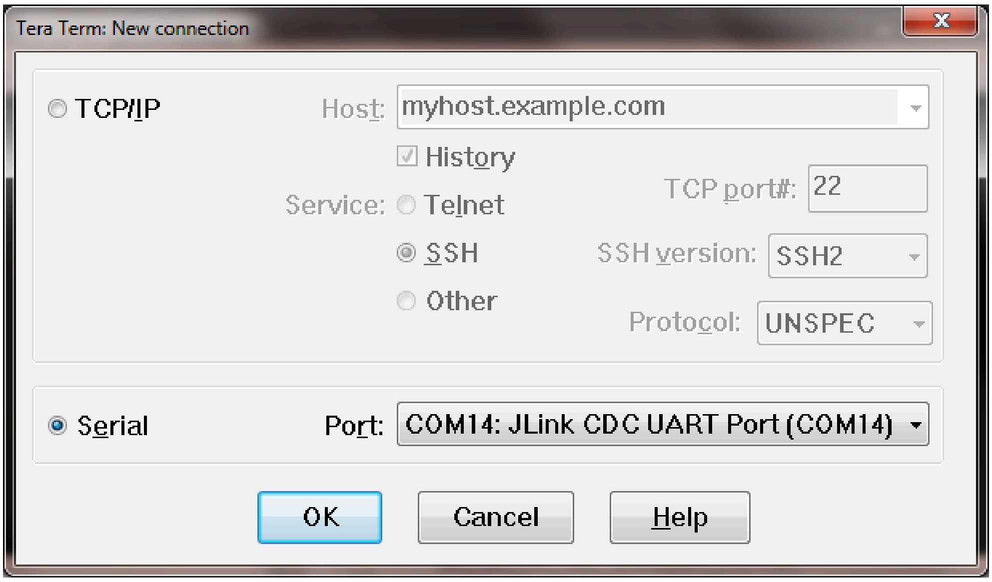


Figure 5. Tera Term – mbed serial port

Connect it to the platform with parameters as shown in Figure 6.

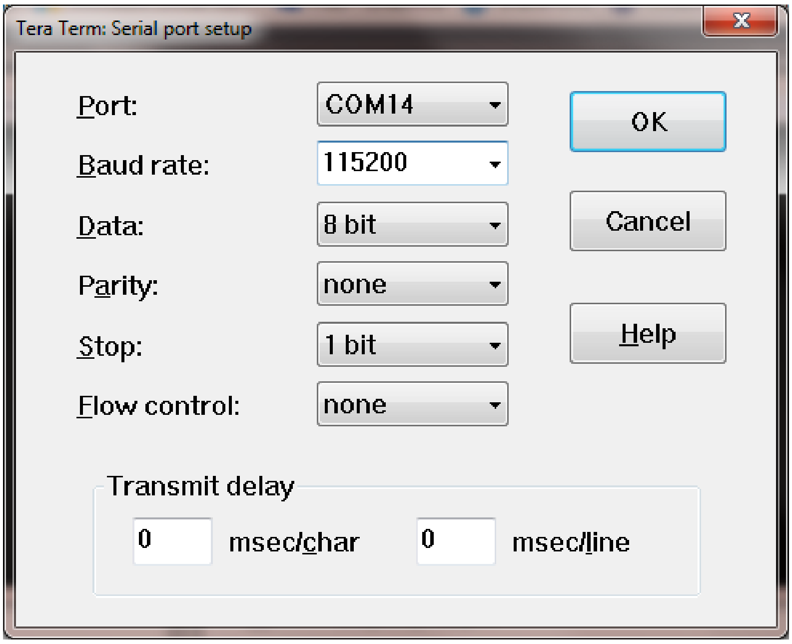


Figure 6. Tera Term – mbed serial port configuration

The start screen is displayed after the board is reset as shown in Figure 7.

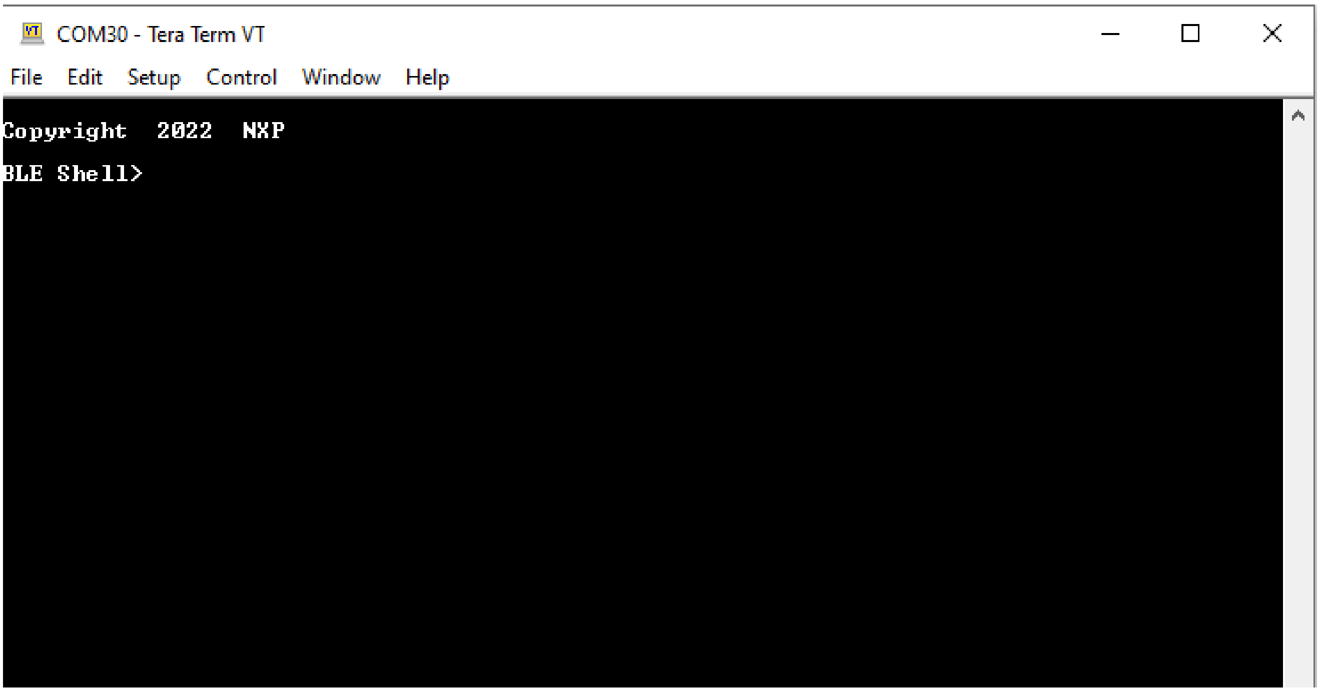


Figure 7. Shell Prompt

4.4 Time client devices

Some applications implement the Current Time Service. To enable this feature, define the `gAppUseTimeService_d` parameter 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. If you connect the device to the phone, the Time Client synchronizes with the phone (pairing and bonding must be active).

5 Bluetooth LE stack and demo applications

5.1 ANCS/AMS client (ancs_c)

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

5.1.1 Implemented profile and services

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

Check the documentation available on the iOS website for details about the ANCS or AMS services, their characteristics, and supported features.

The demo application acts as a GAP Peripheral that advertises a service solicitation for the custom ANCS Service, followed by a solicitation to the AMS Service. It also acts as a GATT Client once connected to a device that offers the ANCS/AMS Service. The application offers some services such as the role of GATT Server.

Once connected to a mobile device offering the ANCS/AMS Service, the application displays information about ANCS Notifications received from that device. This information is followed by the AMS track information (Artist, Album, Title, Duration in seconds). The application also displays the possible remote commands that the device state allows (such as Play, Pause, VolumeUp, VolumeDown). The notifications are received via ATT Notifications, for which the ANCS Client must register on the peer ANCS Server. The same must be done for the AMS server. It initially configures the information that it wants to be notified about. The application also retrieves and displays additional information about the received ANCS notifications. For this purpose, it writes commands to specified characteristics on the ANCS/AMS Server and receives 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 and AMS Service requires Bluetooth LE security to be enabled.

5.1.2 Supported platforms

The ANCS/AMS Client application is supported on the following platform:

- KW45B41Z-EVK
- K32W148-EVK
- FRDM-MCXW71

5.1.3 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/AMS Server (Gap Central) connects to the ANCS/AMS Client (GAP Peripheral), **CONNLED** turns solid. To disconnect, hold the **ADVSW** for 2-3 seconds. The ANCS/AMS Client then re-enters the advertising state.

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

See Table 1 for hardware references.

Table 1. Hardware references

Platform	ADVSW	CONNLED
KW45B41Z-EVK / K32W148-EVK	SW2	LED2
FRDM-MCXW71	SW2	Blue LED
KW47-EVK / MCXW72-EVK	SW2	LED1
FRDM-MCXW72	SW4	Blue LED

5.1.4 Usage

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

- Open a serial terminal application on the PC and connect it to the serial port corresponding to the board on which the ANCS/AMS Client runs. See the details in Section 4.3 "Testing devices", "User Interface". A start screen is displayed immediately after the board is reset. All LEDs must be flashing as shown in Figure 8.

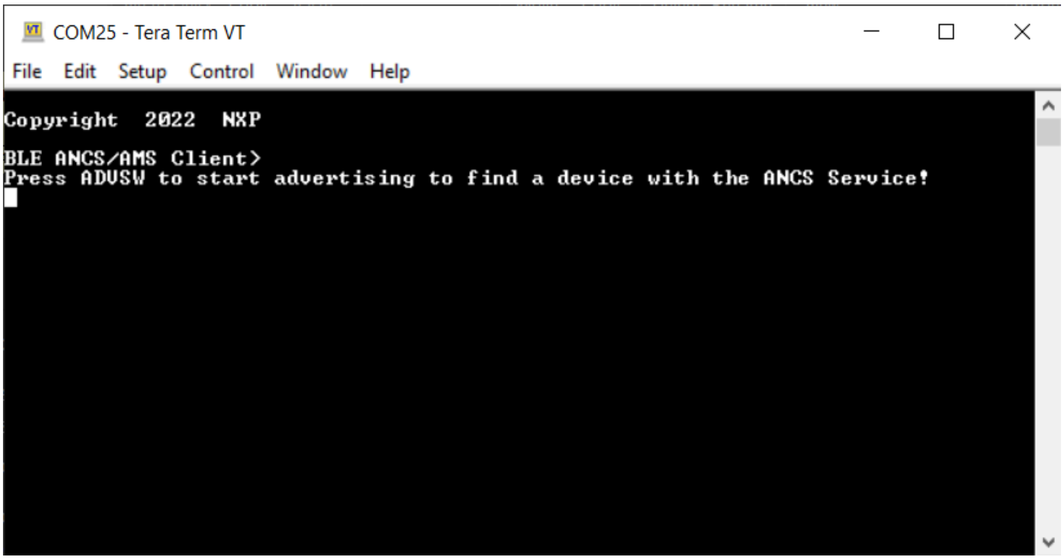


Figure 8. Start screen for ANCS/AMS Client demo application

- Press the **ADVSW** button to start advertising. This instruction is also displayed in the serial terminal as shown in the Figure 8.
- The peer device starts scanning for Bluetooth LE devices and connects to the ANCS/AMS Client device that is advertising.
- Once connected to a peer, the application looks for the ANCS Service and AMS Service and their characteristics. If they are found, the ANCS Client tries to register for receiving notifications and AMS for the tracking data and commands. See Figure 9.
- If any security-related ATT errors are encountered, then the application automatically performs Pairing and Bonding and retries the failed ATT operations. Depending on the negotiated pairing method, user interaction might be needed to complete the Pairing. Follow the onscreen instructions provided by both the ANCS/AMS Client and the mobile device. If the ANCS/AMS Client generates a passkey, then the default 999999 passkey is used.

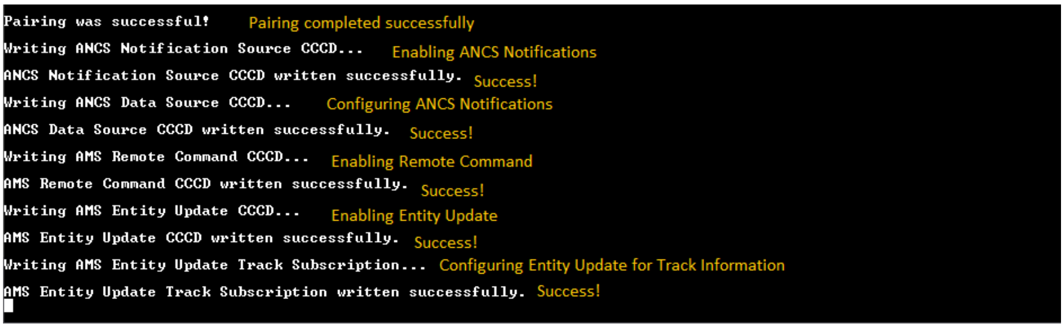


Figure 9. Pairing is successful message

- After bidirectional communication is established via GATT, the ANCS/AMS Client starts displaying ANCS Notifications information as shown in Figure 10 below. The combination of the two services (ANCS and AMS) is shown in Figure 11 and Figure 12. If no media is playing and no player is active, the serial interface looks as shown in Figure 11. When media is playing, a player is active and the state must look similar to Figure 12.

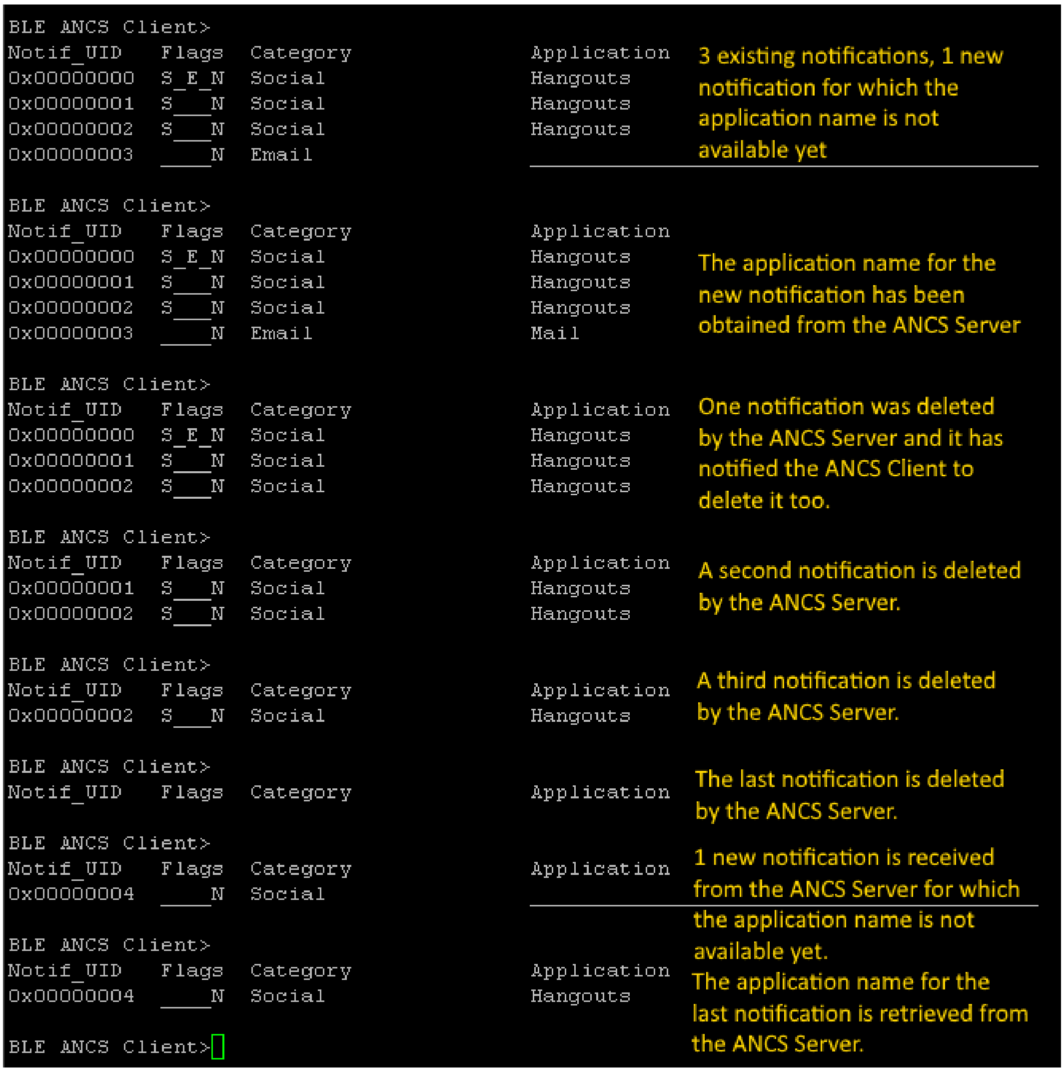


Figure 10. ANCS notification information

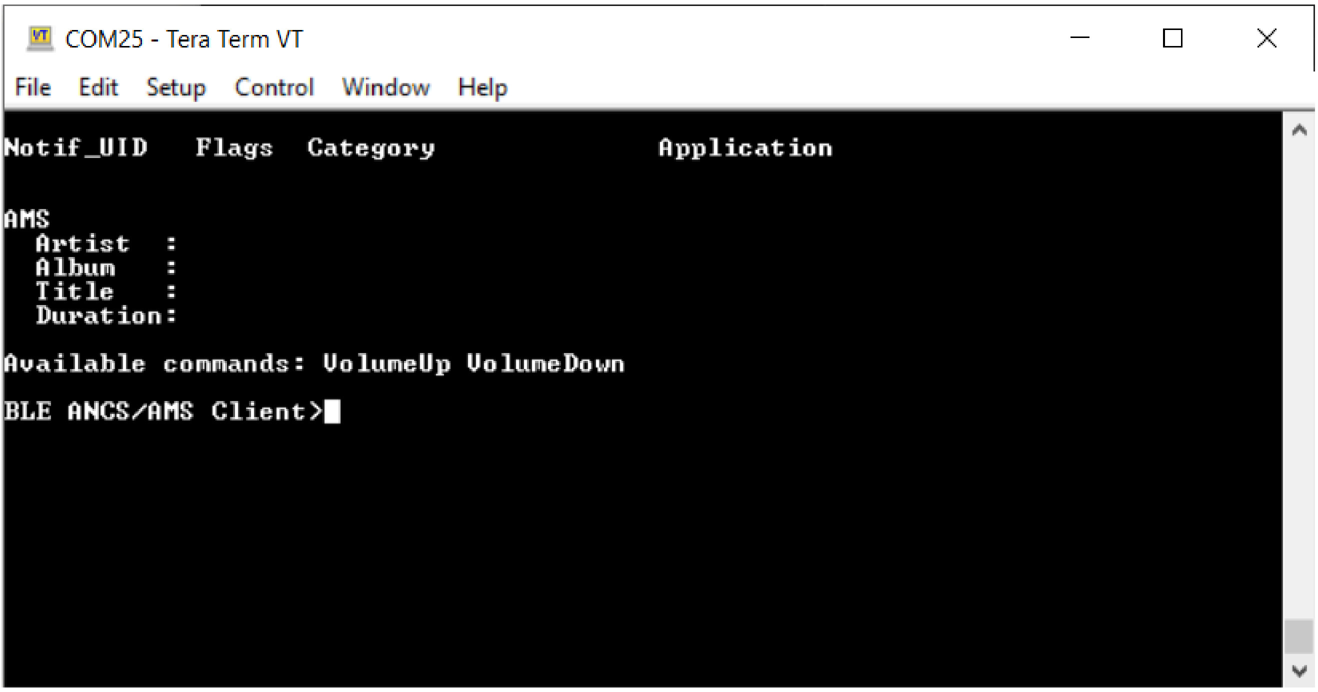


Figure 11. AMS no active player

Figure 12. AMS with player active

5.2 Beacon

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

5.2.1 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.

5.2.2 Supported platforms

The following platforms support the Beacon application:

- KW45B41Z-EVK
- K32W148-EVK
- FRDM-MCXW71
- KW47-EVK
- MCXW72-EVK
- FRDM-MCXW72

5.2.3 User interface

After flashing the beacon, the sensor is put in deep sleep (all LEDs are off). To flash the board in case the beacon is put in Deep-sleep mode, press the **ADVSW** or **RESET** button. After this step, any attached debugger loses its connection. The default configuration of the application enables low power, which disables LED support. The user can manually change the configuration and enable LED support, otherwise all subsequent LED behavior references are ignored.

By default, the application uses extended advertising. However, it can be configured to use legacy advertising by setting the `gBeaconAE_c` define to `0`. In the legacy configuration, the first press of the advertising switch starts the legacy advertising and the second press stops it.

The Table 2 lists details of the hardware references.

Table 2. Hardware references

Platform	ADVSW	ADVLED	EXTADVLED
KW45B41Z-EVK / K32W148-EVK	SW2	LED2	LED1
FRDM-MCXW71	SW2	Blue LED	RGB LEDS
KW47-EVK / MCXW72-EVK	SW2	LED2	LED1
FRDM-MCXW72	SW4	Blue LED	RGB LEDS

5.2.4 Usage

The beacon can be tested with any Bluetooth® Smart Ready products available on the market. The IoT Toolbox can also be used to showcase the profile functionality, as shown in Figure 13.

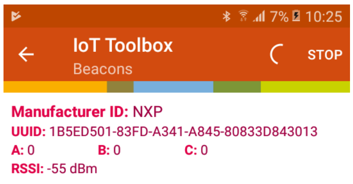


Figure 13. IoT Toolbox Beacon Demo

5.2.5 Beacon usage with extended advertising

To use the Beacon application with the advertising extensions capabilities, the `gBeaconAE_c define` option 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 first **ADVSW** press starts legacy advertising, **CONNLED** turns solid.
- The second **ADVSW** press stops legacy advertising and starts extended advertising, **CONNLED** turns off, **EXTADVLED** turns solid.
- The third **ADVSW** press stops extended advertising carrying data and then starts extended advertising without data and periodic advertising, **EXTADVLED** starts flashing.
- The fourth **ADVSW** press stops periodic advertising and extended advertising without data and starts legacy advertising and extended advertising, both **CONNLED** and **EXTADVLED** turn solid.
- The fifth ADVSW press stops them all, both **CONNLED** and **EXTADVLED** turn off.

Note: *Periodic advertising support is currently disabled at Link Layer level.*

Not all smartphones support extended advertising, hence a different method to view the AE beacon is to use the `ble_shell` application. In order to do this, perform the following steps:

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 **ADVSW** 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. See Figure 14.

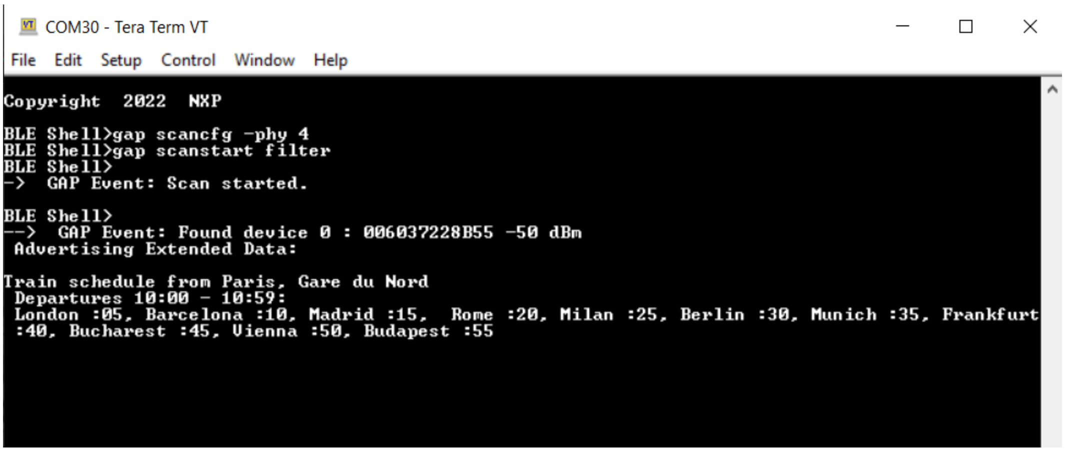


Figure 14. Shell commands to view extended advertising

5. To start the periodic advertising, press **ADVSW** button again on the beacon.
   Note: *Periodic advertising support is currently disabled at Link Layer level.*
6. To sync with the beacon, issue the following commands on the shell terminal as shown in Figure 15.

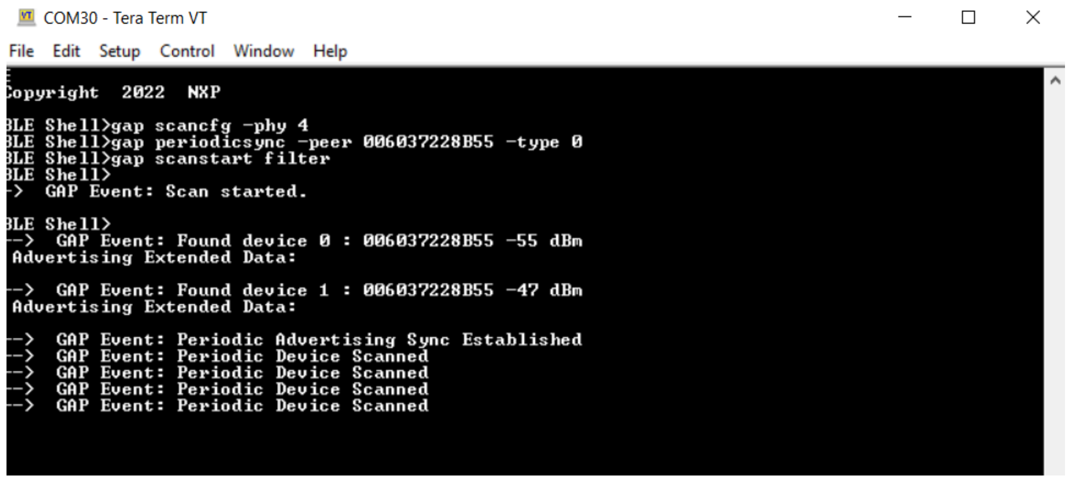


Figure 15. Shell commands to view the periodic advertising

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

5.2.5.1 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 **ADVSW** 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. See Figure 16.

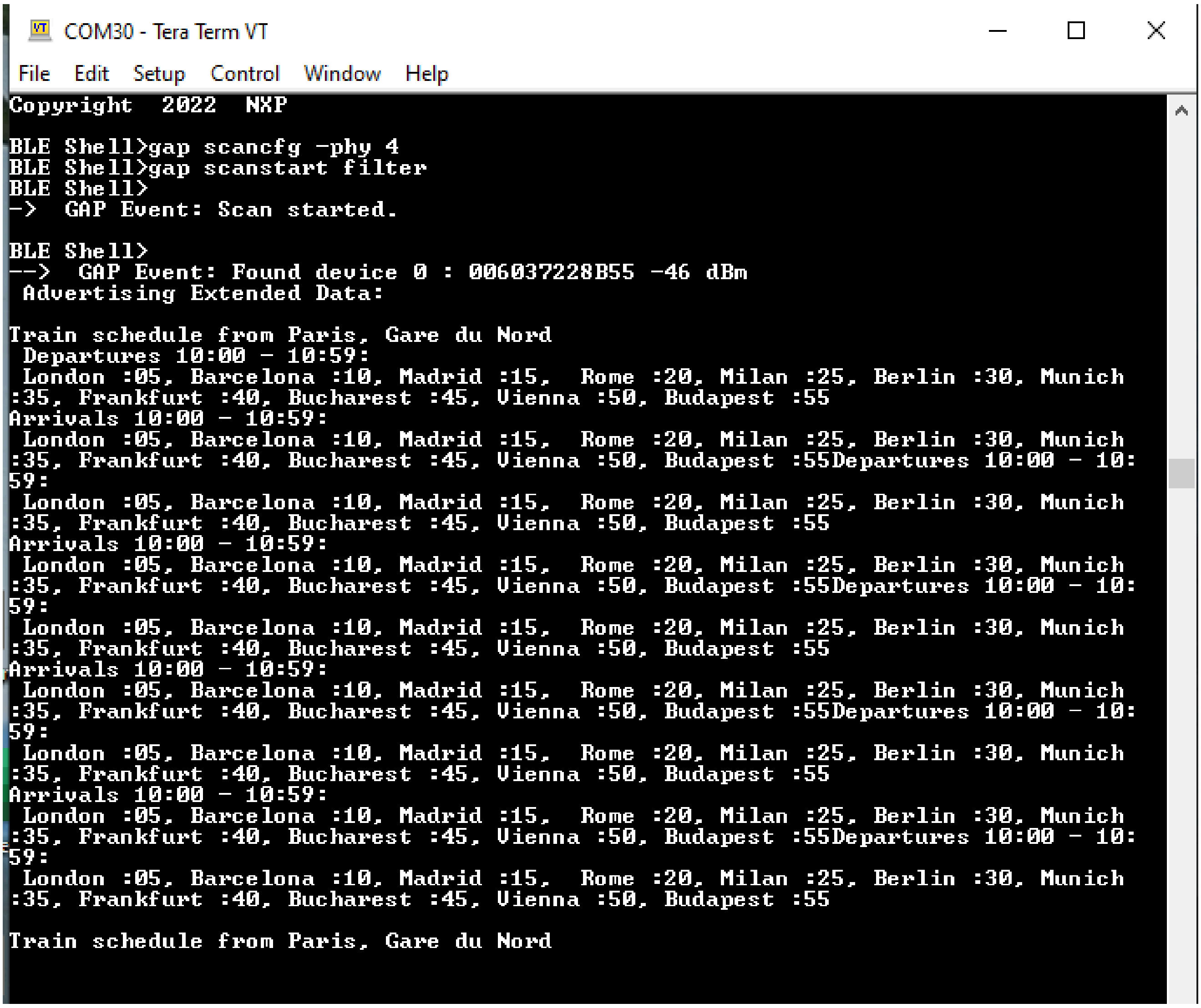


Figure 16. Shell command to view extended advertisements with large data (kw45)

5.3 Bluetooth LE FSCI Black Box

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

5.3.1 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 using a custom application that supports the FSCI protocol and commands.

5.3.2 Supported platforms

The following platforms support Bluetooth LE FSCI Black Box application:

- KW45B41Z-EVK
- K32W148-EVK
- FRDM-MCXW71
- KW47-EVK
- MCXW72-EVK
- FRDM-MCXW72

5.3.3 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. See Figure 17. The serial communication parameters are: baud rate 115200, 8N1, and no flow control.

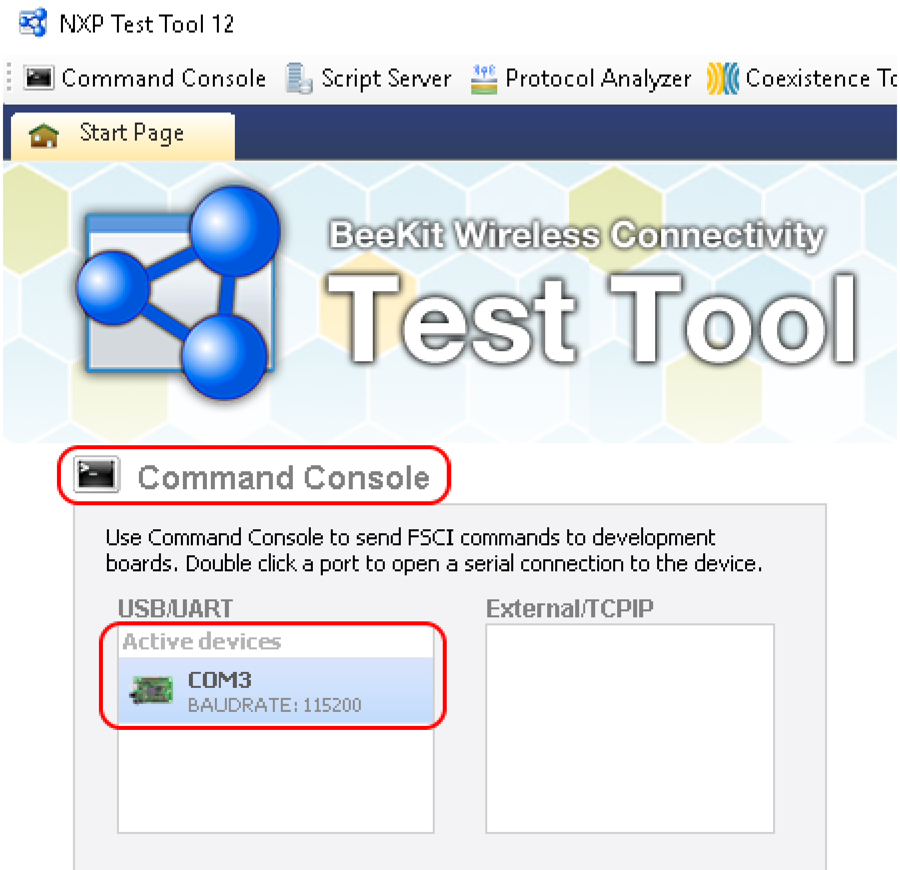


Figure 17. Test tool command console serial port selection

4. Select the appropriate Test Tool XML file from the drop-down list for the release being used and send commands to the application. An example is shown in Figure 18.

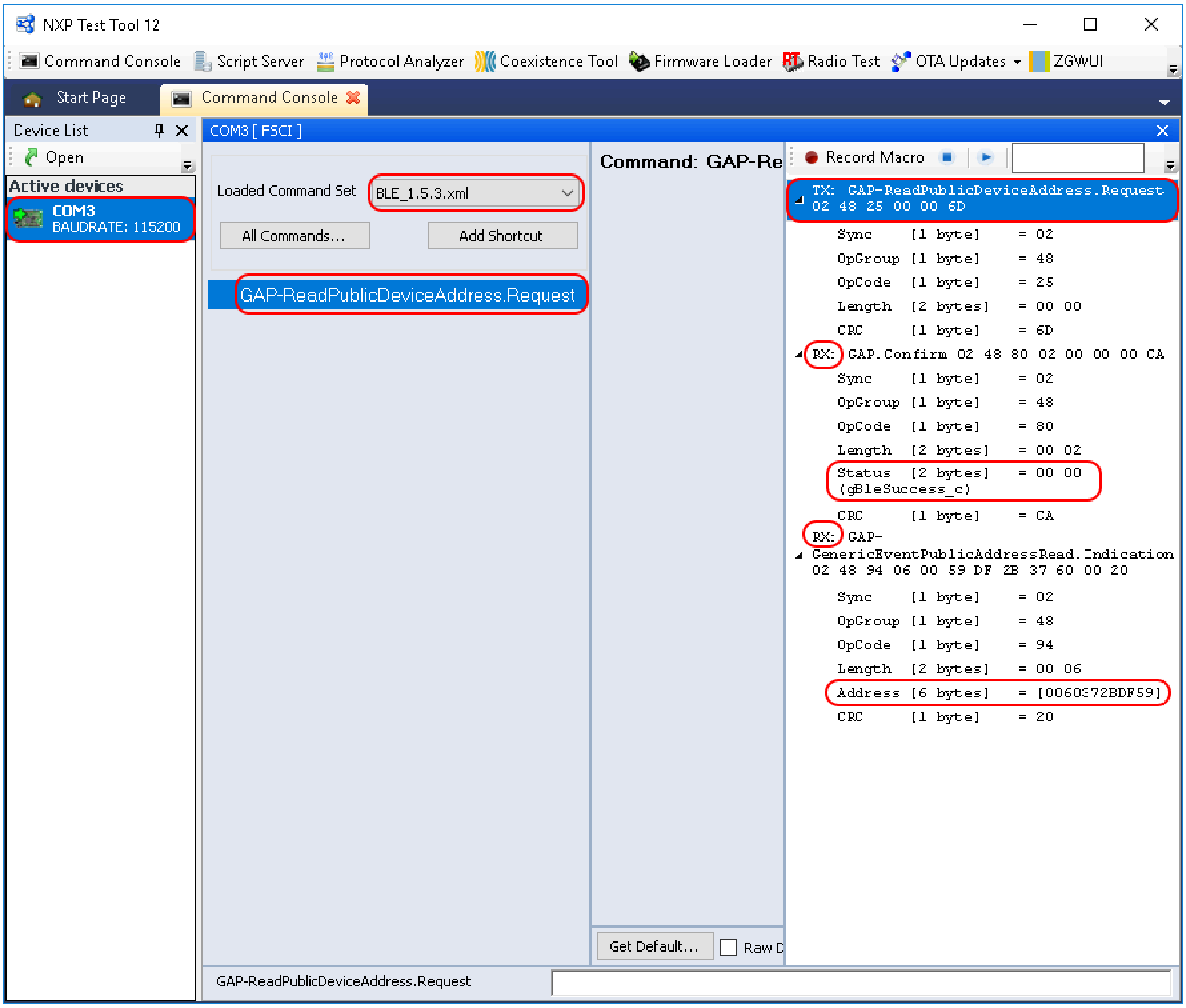


Figure 18. FSCI black box command example

5.4 EATT Central

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

5.4.1 Implemented profiles and services

The EATT Central application implements a GATT server and the following profiles and services:

- Generic Attribute Profile

The application behaves as a GAP central node. It searches for an EATT peripheral to connect to. After connecting, it performs service discovery, initiates an EATT connection and configures indications on the peripheral for services A and B. The Central reports the received service data and the steps taken during the setup on a terminal connected to an UART port.

5.4.2 Supported platforms

The EATT Central application is supported on the following platforms:

- KW45B41Z-EVK
- K32W148-EVK
- FRDM-MCXW71
- KW47-EVK
- MCXW72-EVK
- FRDM-MCXW72

5.4.3 User interface

After flashing the board, the central is in Idle mode (all LEDs flashing). To start scanning, press the **SCANSW** button. After connecting to the peripheral, the **CONNLED** turns solid. The data information together with the bearer it was received on is sent over UART. To disconnect the node, hold the **SCANSW** button pressed for 2-3 seconds. The node then restarts scanning.

See Table 3 for hardware references.

Table 3. Hardware references

Platform	SCANSW	CONNLED
KW45B41Z-EVK / K32W148-EVK	SW2	LED2
FRDM-MCXW71	SW2	Blue LED
KW47-EVK / MCXW72-EVK	SW2	LED2
FRDM-MCXW72	SW4	Blue LED

5.4.4 Usage

The application can be tested using another board flashed with the EATT Central application as described in the Section 5.5 "EATT Peripheral".

1. Open a serial port terminal and connect it to board, in the same manner described in Section 4.3 "Testing devices". The start screen is displayed after the board is reset.
2. To start scanning for devices, press the **SCANSW** button on the EATT Central board. To make it enter discoverable mode, perform the same step on the EATT Peripheral board. The host connects with the board after it sees it advertise the service A and service B UUIDs. After connecting, the central performs service

discovery, indicates its EATT support to the server by writing the Client Supported Features characteristic, enables indications for services A and B, and then initiates an EATT connection with the server. See Figure 19.

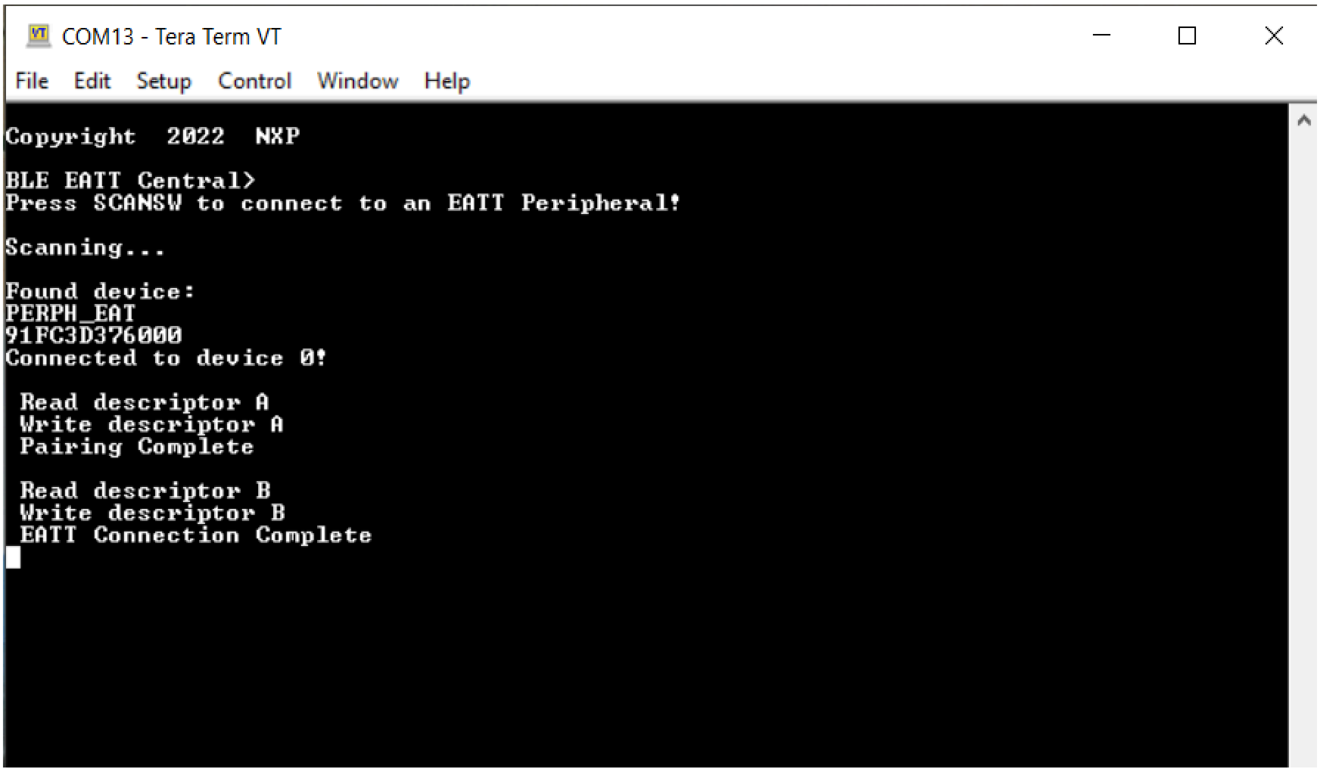


Figure 19. Output console on the EATT Central connected with an EATT Peripheral

3. After the EATT connection is complete, the console displays the received data and the bearer on which it was sent, as shown in Figure 20.

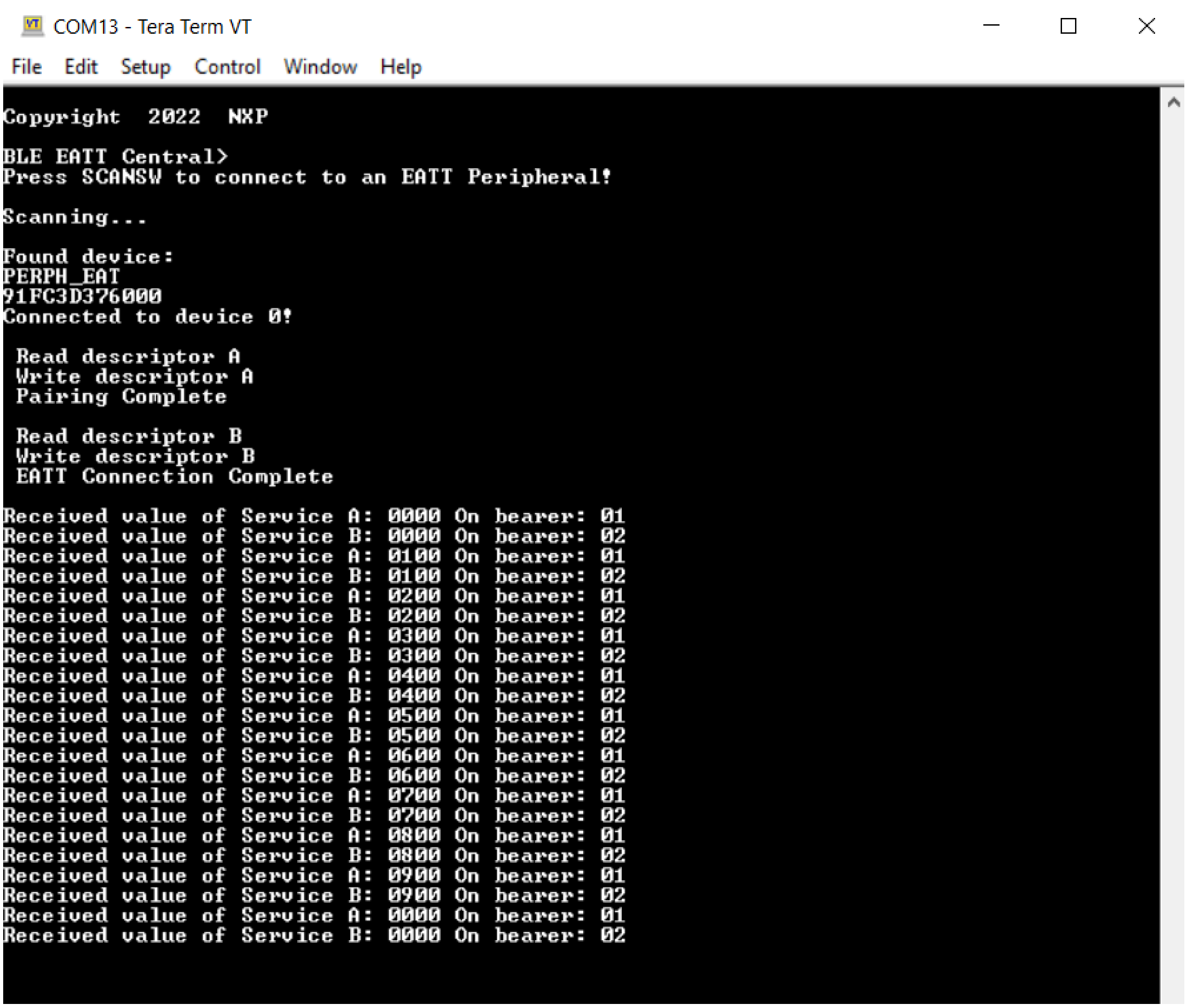


Figure 20. Output console on EATT Central

5.5 EATT Peripheral

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

5.5.1 Implemented profiles and services

The EATT Peripheral application implements a GATT server and the following profiles and services:

- Battery Service v1.0
- Device Information Service v1.1
- Generic Attribute Profile

The Generic Attribute Profile includes the Server Supported Features characteristics, which is used to indicate EATT support to the peer, and the Client Supported Features characteristic, which is used by the peer to indicate its own EATT support.

The application behaves as a GAP peripheral node. It enters GAP General Discoverable Mode and waits for a GAP central node to connect. The application implements two custom services, Service A and Service B. After the EATT connection is completed, the peer must enable indications for the two services to periodically receive profile data over EATT.

5.5.2 Supported platforms

The EATT Peripheral application is supported on the following platforms:

- KW45B41Z-EVK
- K32W148-EVK
- FRDM-MCXW71
- KW47-EVK
- MCXW72-EVK
- FRDM-MCXW72

5.5.3 User interface

After flashing the board, the peripheral 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. The node then waits for an EATT connection request and for the client to configure indications for the two services. The service information is sent over enhanced bearers only. The values indicated are cycled between 0 and 10. To disconnect the node, hold the ADVSW button for 2-3 seconds. The node then re-enters GAP Discoverable Mode and starts advertising.

Table 4 details below the hardware references.

Table 4. Hardware references

Platform	ADVSW	CONNLED
KW45B41Z-EVK / K32W148-EVK	SW2	LED2
FRDM-MCXW71	SW2	Blue LED
KW47-EVK / MCXW72-EVK	SW2	LED2
FRDM-MCXW72	SW4	Blue LED

5.5.4 Usage

The application can be tested using another board flashed with the EATT Central application as described in the EATT Central Section.

5.6 HCI Black Box

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

5.6.1 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. Refer to 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. The Command Console application can be downloaded from the NXP website. Alternatively you can also download it using a custom application that supports the HCI protocol and commands and events over serial interface.

Note: *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 the article "FRDM-KW40Z Bluetooth LE Controller Usage with the Linux hcitool".

5.6.2 Supported platforms

The following platforms support the HCI Black Box application:

- KW45B41Z-EVK
- K32W148-EVK
- FRDM-MCXW71
- KW47-EVK
- MCXW72-EVK
- FRDM-MCXW72

5.6.3 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 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 HCI Black Box application runs. The serial communication parameters are: baud rate 115200, 8N1, and no flow control. See Figure 21.

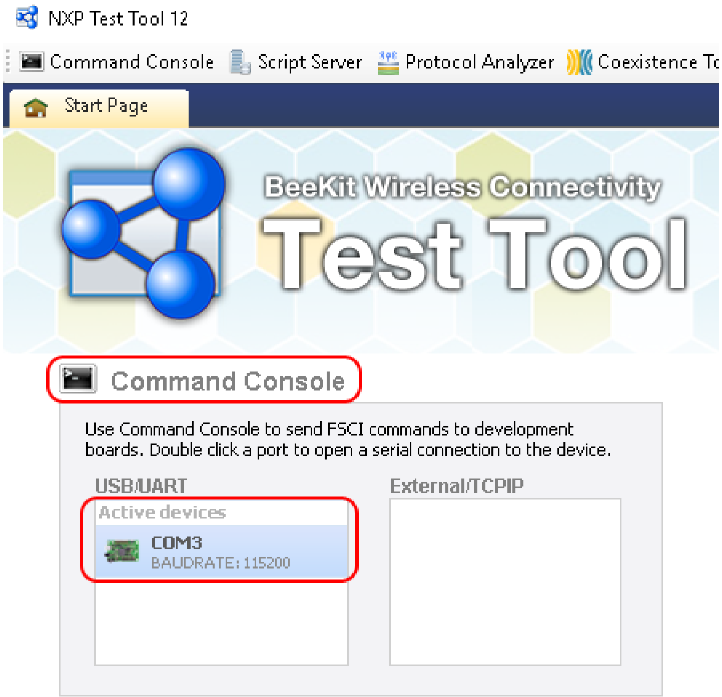


Figure 21. Test tool command console serial port selection

4. Select the appropriate Test Tool HCI XML file from the drop-down list for the release you are using. Send a few commands to the application. An example is shown in Figure 22.

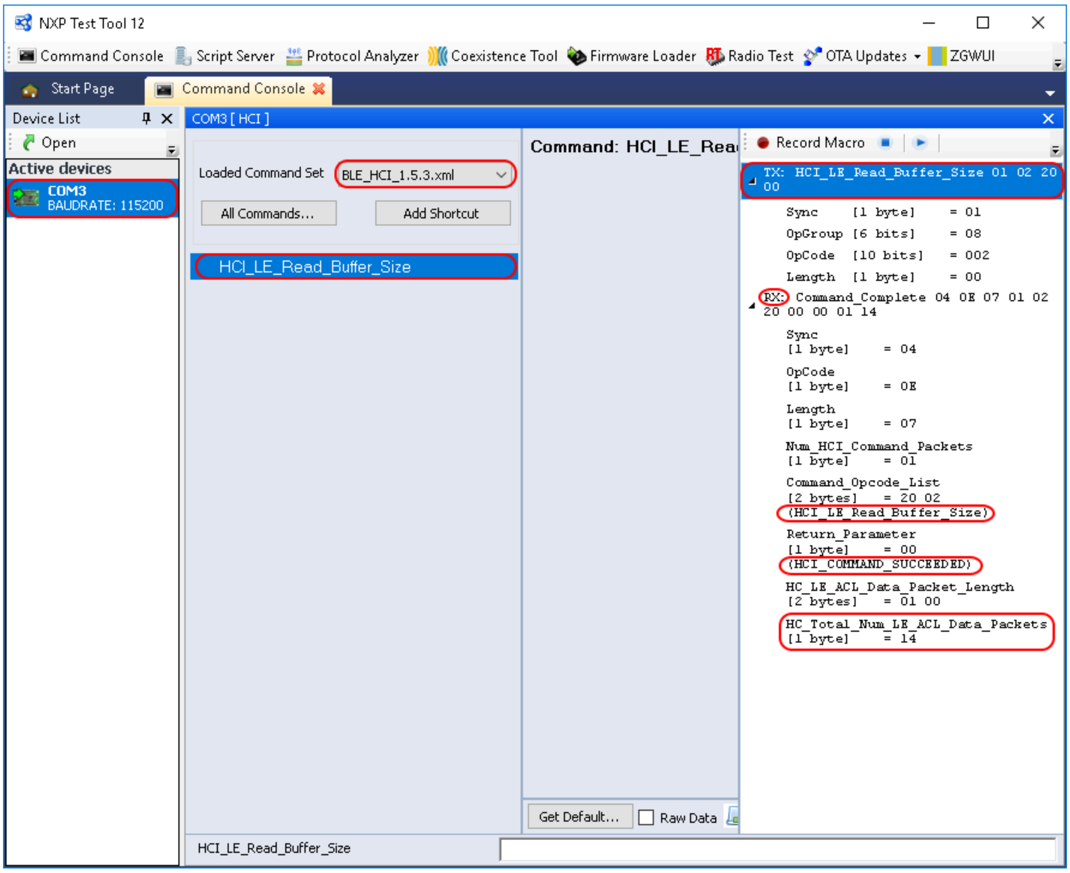


Figure 22. HCI black box command example

5.7 HID Device (Mouse)

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

5.7.1 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.

5.7.2 Supported platforms

The following platforms support the HID Device application:

- KW45B41Z-EVK
- K32W148-EVK
- FRDM-MCXW71
- KW47-EVK
- MCXW72-EVK
- FRDM-MCXW72

5.7.3 User interface

After flashing the board, the device enters Idle mode with all LEDs flashing. To start advertising, press the **ADVSW** button. 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.

Table 5 details below the hardware references.

Table 5. Hardware references

Platform	ADVSW	CONNLED
KW45B41Z-EVK / K32W148-EVK	SW2	LED2
FRDM-MCXW71	SW2	Blue LED
KW47-EVK / MCXW72-EVK	SW2	LED2
FRDM-MCXW72	SW4	Blue LED

5.7.4 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. See Figure 23. 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.

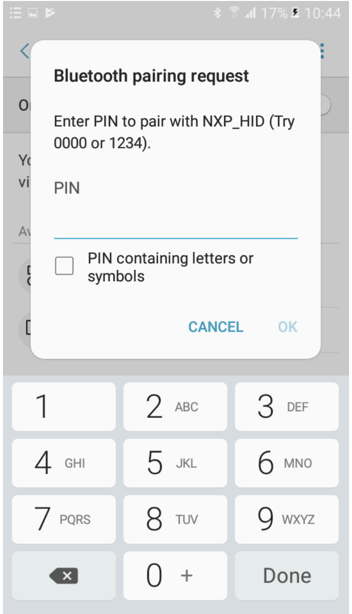


Figure 23. Enter PIN prompt on Android platform

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 as shown in Figure 24.

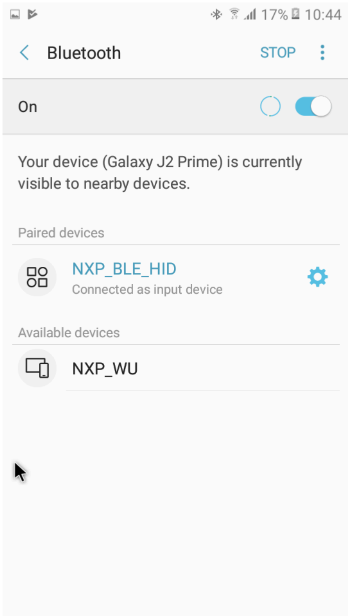


Figure 24. HID Mouse detected by Android platform

5.8 HID Host

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

5.8.1 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.

5.8.2 Supported platforms

The following platforms support the HID Host application:

- KW45B41Z-EVK
- K32W148-EVK
- FRDM-MCXW71
- KW47-EVK
- MCXW72-EVK
- FRDM-MCXW72

5.8.3 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 Section 5.8.3 "User interface" below for hardware references.

Table 6. Hardware references

Platform	SCANSW	CONNLED
KW45B41Z-EVK / K32W148-EVK	SW2	LED2
FRDM-MCXW71	SW2	Blue LED
KW47-EVK / MCXW72-EVK	SW2	LED2
FRDM-MCXW72	SW4	Blue LED

5.8.4 Usage

The application is built to work only with the HID Device application presented in Section 5.7 "HID Device (Mouse)". 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 4.3 "Testing devices". The start screen is displayed after the board is reset.
2. To start scanning for devices, press the **SCANSW** button on the HID Host board. To make it enter discoverable mode, perform the same step on the HID device board. 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 in Figure 25.

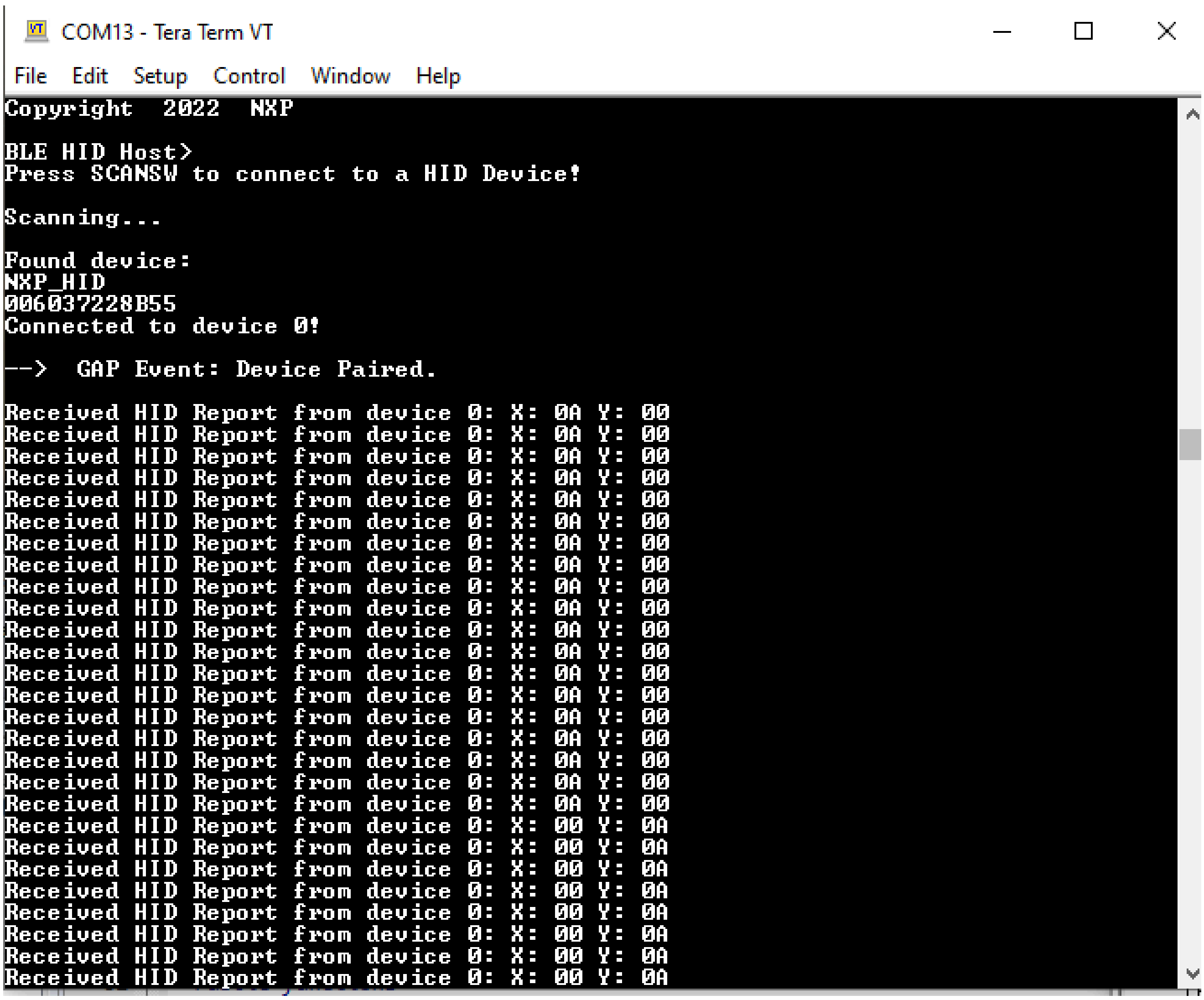


Figure 25. HID host

3. 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 displays reports from both devices, as shown in Figure 26.

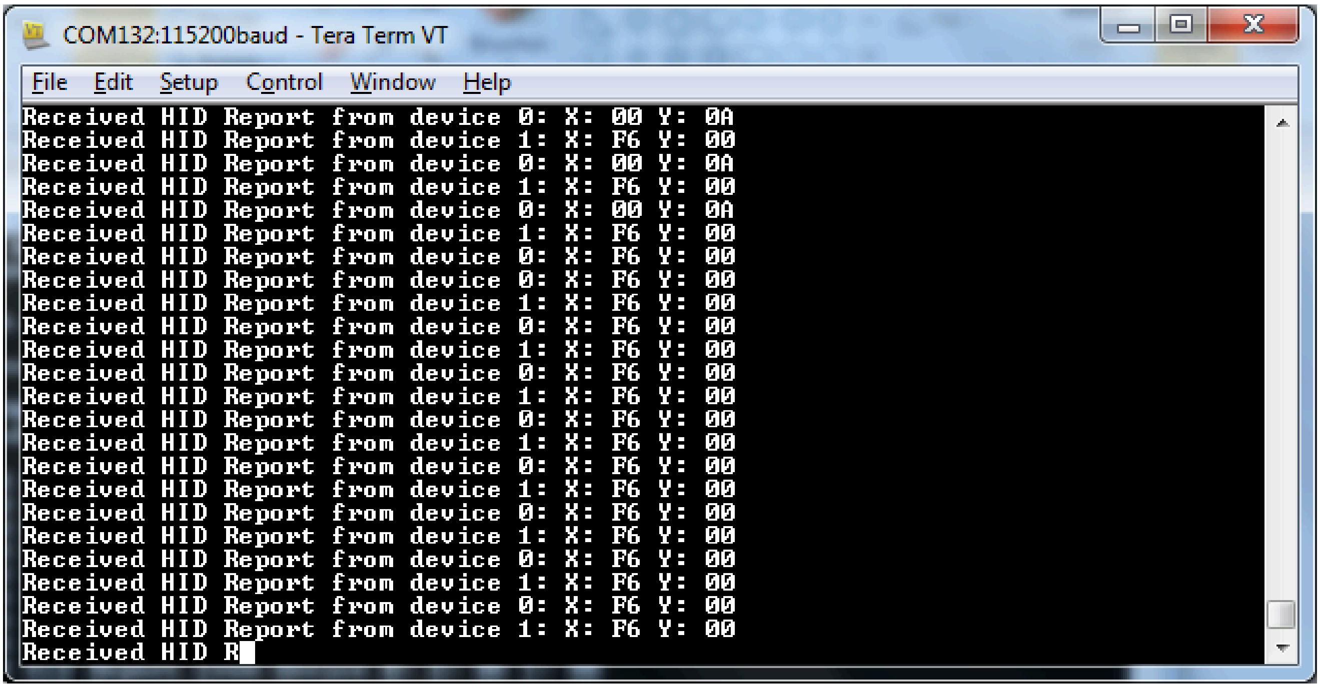


Figure 26. Tera Term – Output Console on HID Host with 2 peripherals connected

5.9 Low-power temperature sensor and collector

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

5.9.1 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.

5.9.2 Supported platforms

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

- KW45B41Z-EVK
- K32W148-EVK
- FRDM-MCXW71
- KW47-EVK
- MCXW72-EVK
- FRDM-MCXW72

5.9.3 User interface

After flashing the board, both nodes enter Low-power mode. In case the sensor is put in deep sleep, press WAKESW or RESET. To flash the board in case the sensor is put in deep sleep, press either **WAKESW** or **RESET** button. By default, the application is configured to be in low power mode, which disables LED support.

The user can manually change this configuration and enable LED support, else all subsequent LED behavior references are ignored and all LEDs are off. The devices disconnect and enter Deep-sleep only if low power is enabled. When the node is awake and communicating, **CONNLED** is on. To wake up the node, press the **WAKESW** button.

See Table 7 below for hardware references.

Table 7. Hardware references

Platform	WAKESW	CONNLED
KW45B41Z-EVK / K32W148-EVK	SW2	LED2
FRDM-MCXW71	SW2	Blue LED
KW47-EVK/MCXW72-EVK	SW2	LED2
FRDM-MCXW72	SW4	Blue LED

5.9.4 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 Section 4.3 "Testing devices". 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 Deep-sleep mode, 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 Deep-sleep mode 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 Deep-sleep mode, 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 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 Deep-sleep mode. 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 2M, if both the sensor and the collector support this feature as shown in Figure 27. The PHY update is configurable from the application.

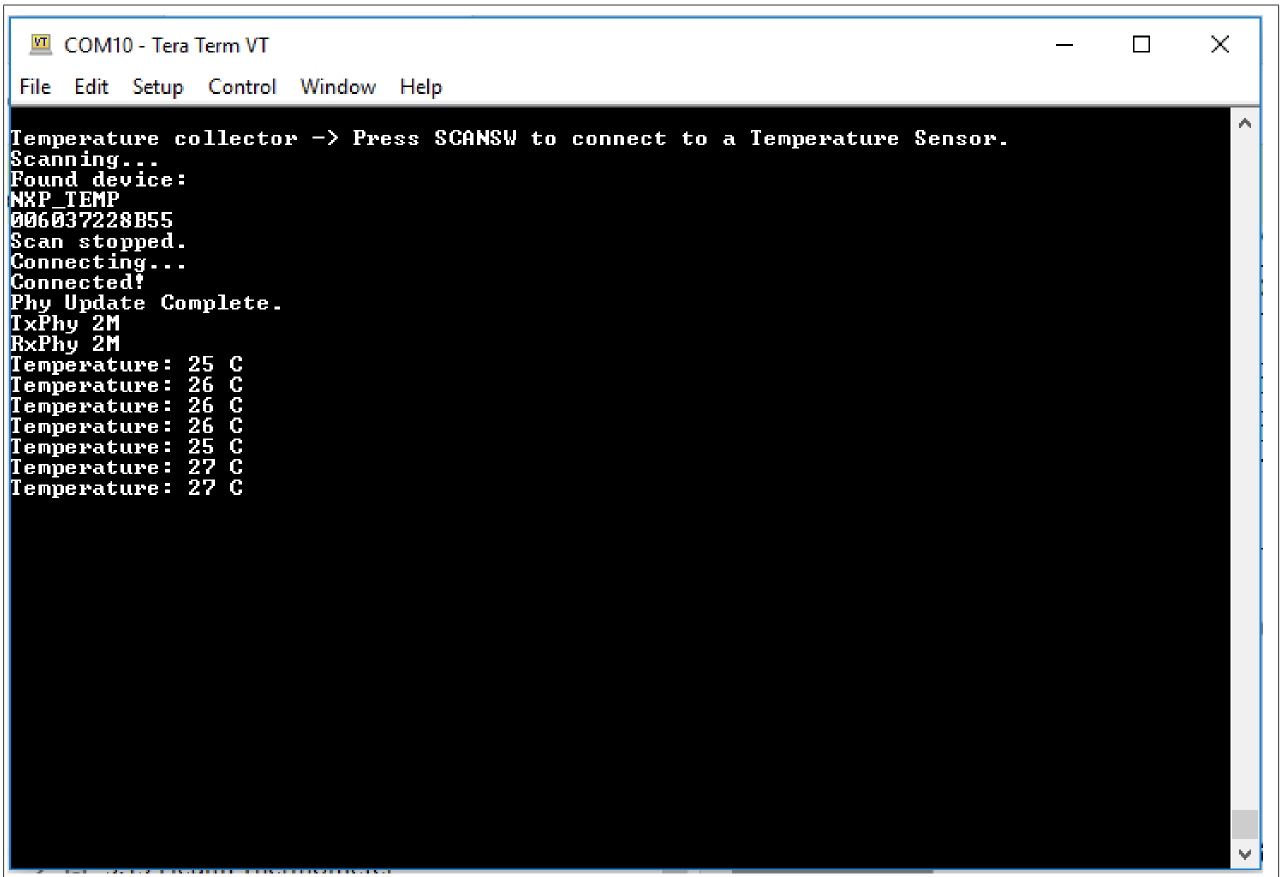


Figure 27. Output Console on Temperature Collector

5. 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 Deep-sleep mode.

5.10 Low-power extended advertising Peripheral and Central

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

5.10.1 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.

5.10.2 Supported platforms

The following platforms support Extended Advertising Peripheral and Central applications:

- KW45B41Z-EVK
- K32W148-EVK
- FRDM-MCXW71
- KW47-EVK
- MCXW72-EVK
- FRDM-MCXW72

Deep-sleep mode is used by default.

5.10.3 User interface

After flashing the board, both nodes enter Deep-sleep mode. To flash the board again, press **WAKESW**. The application default configuration enables low power that disables LED support. The user disables low power and enables LED support setting to 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 Table 8 and Table 9 below for hardware references.

Hardware references:

Table 8. Extended Advertising Peripheral

Platform	WAKESW	OPTSW	ADVLED	CONNLED
KW45B41Z-EVK / K32W148-EVK	SW3	SW2	LED2	LED1
FRDM-MCXW71	SW4	SW2	Blue LED	RGB LED
KW47-EVK / MCXW72-EVK	SW3	SW2	LED2	LED1
FRDM-MCXW72	SW2	SW4	Blue LED	RGB LED

Table 9. Extended Advertising Central

Platform	WAKESW	SCANLED	CONNLED
KW45B41Z-EVK / K32W148-EVK	SW2	LED2	LED1
FRDM-MCXW71	SW2	Blue LED	RGB LED
KW47-EVK / MCXW72-EVK	SW3	LED2	LED1
FRDM-MCXW72	SW4	Blue LED	RGB LED

5.10.4 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 as shown in Figure 28 and Figure 29.

Figure 28. adv_ext_peripheral start screen

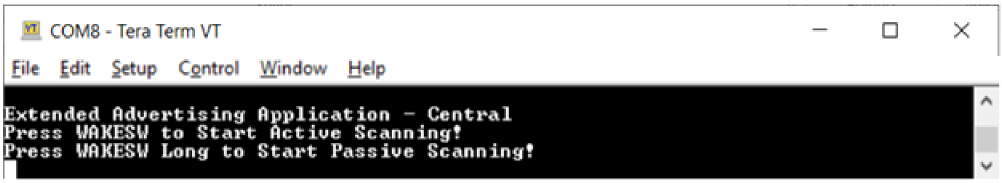


Figure 29. adv_ext_central start screen

2. On the board that implements the adv_ext_peripheral application, press the **WAKESW** button. The board exits Deep-sleep mode and displays the menu as shown in Figure 30.

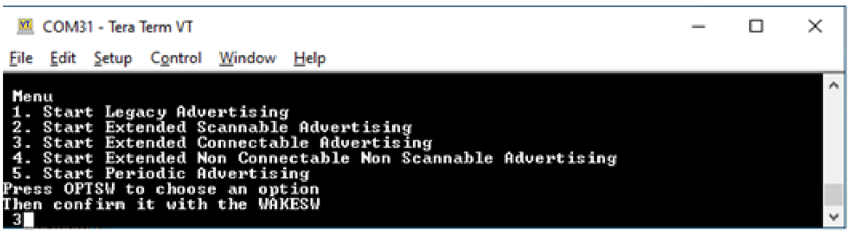


Figure 30. Choosing a menu option on adv_ext_peripheral

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 Starts Extended Connectable

Advertising), press the **WAKESW** again to make a decision. The advertising type chosen is started 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 advertising is off and all connections are terminated, the board enters low-power mode until the **WAKESW** button is pressed again.
When `gAppLowpowerEnabled_d` 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.

3. On the board that implements the **adv_ext_central** application, there are 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, select active scanning. The device wakes up, starts scanning, and enters Deep-sleep mode. The scanning ends when the 60 seconds timeout is reached or when a connection with an adv_ext_peripheral device is established.
During scanning, all advertisements caught from adv_ext_peripheral devices are displayed on the terminal window as shown in Figure 31. When an extended non-connectable, non-scannable 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 Deep-sleep mode until the WAKESW is pressed again and all syncs with periodic advertising trains are terminated. If `gAppLowpowerEnabled_d` 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.
Note: *Periodic advertising support is currently disabled at Link Layer level.*

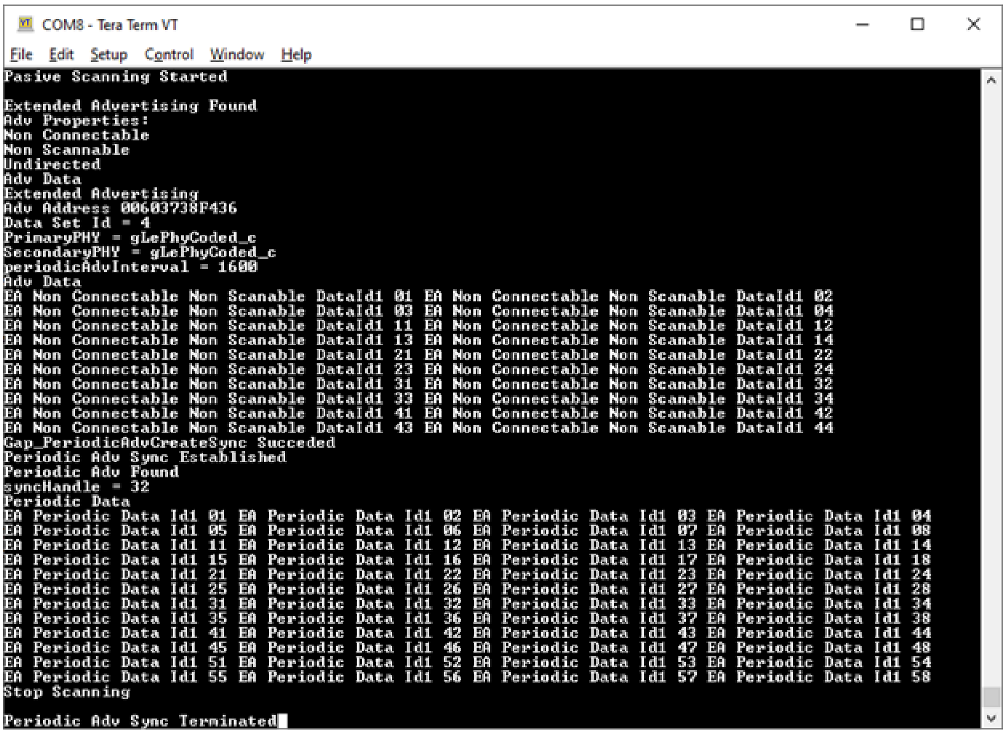


Figure 31. Advertising caught on adv_ext_central console

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 Deep-sleep mode. The peripheral sends a notification with the value of the temperature read through an ADC from a thermistor, if present, or randomly generated, if not. When the central receives the notification, it displays it on the terminal window and disconnects in 5 seconds as shown in Figure 32.

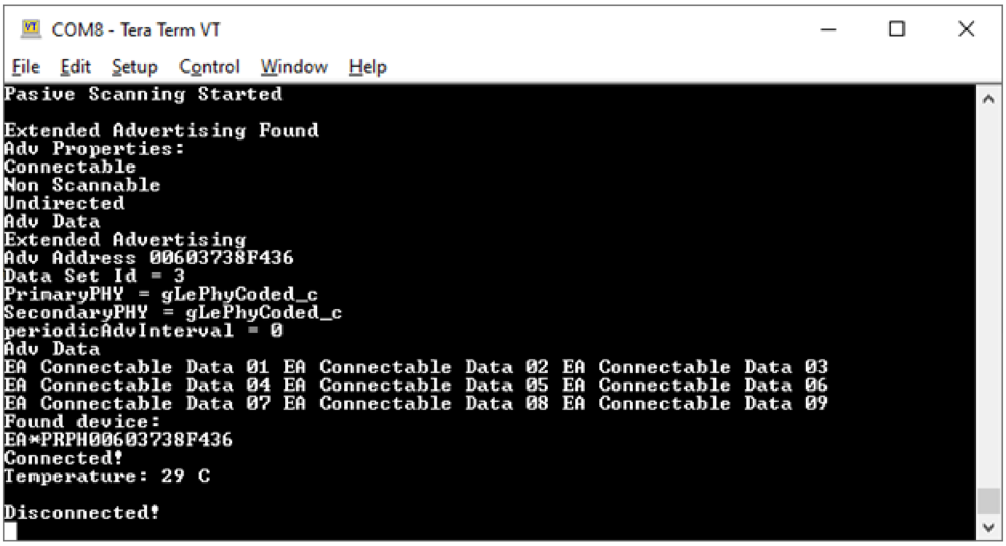


Figure 32. Connection on adv_ext_central console

5.11 Over the Air Programming (OTAP)

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

5.11.1 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 3 applications: an OTAP Client embedded application, an OTAP Server embedded application, and an Over the Air Programming PC application. The OTAP Client embedded application has two versions, an ATT version and a L2CAP version each using a different transfer method.

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 Over the Air Programming 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.

5.11.2 Supported platforms

The following platforms support the OTAP applications:

- KW45B41Z-EVK
- K32W148-EVK
- FRDM-MCXW71
- KW47-EVK
- MCXW72-EVK
- FRDM-MCXW72

5.11.3 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.

Start the OTAP Server PC application after the embedded applications are flashed to the boards. The application creates an OTAP image file using the provided executable `.srec` or `.bin` file. It then connects to the embedded OTAP Server via the configured serial interface and waits for commands. The application shows details about the image file creation and allows the OTAP upgrade image file header to be configured. The log view of the application displays the interactions between the OTAP Client and the OTAP Server.

See Table 10 for the hardware references.

Table 10. Hardware references

Platform	ADVSW	SCANSW	CONNLED
KW45B41Z-EVK / K32W148-EVK	SW2	SW2	LED2
FRDM-MCXW71	SW2	SW2	Blue LED
KW47-EVK / MCXW72-EVK	SW2	SW2	LED2
FRDM-MCXW72	SW4	SW4	Blue LED

5.11.4 Usage with Over The Air Programming Tool

Below is a list of requirements for usage with Test Tool for Connectivity products:

- Over The Air Programming Tool 1.4.0 or newer on CONNECTIVITY-TOOL-SUITE
- Serial COM port drivers – these are board-specific.

To run the application, follow the steps below:

1. Flash the OTAP Server onto a supported platform 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.
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 in Figure 33.
   When compiling an image for the Over-the-Air update, the `gEraseNVMLink_d` linker symbol must be set to `0` and `gUseSecureBoot_d` set to `1` only if you are using external storage support.
   In a specific use case, external storage support might be used and the image is created in MCUXpresso IDE. If this image is close to the maximum size of the internal storage, then a new flash section must be added before the NVM section. This step is necessary to ensure that the signature data does not overlap the NVM section. The flash section should be significant enough to accommodate the signature data.

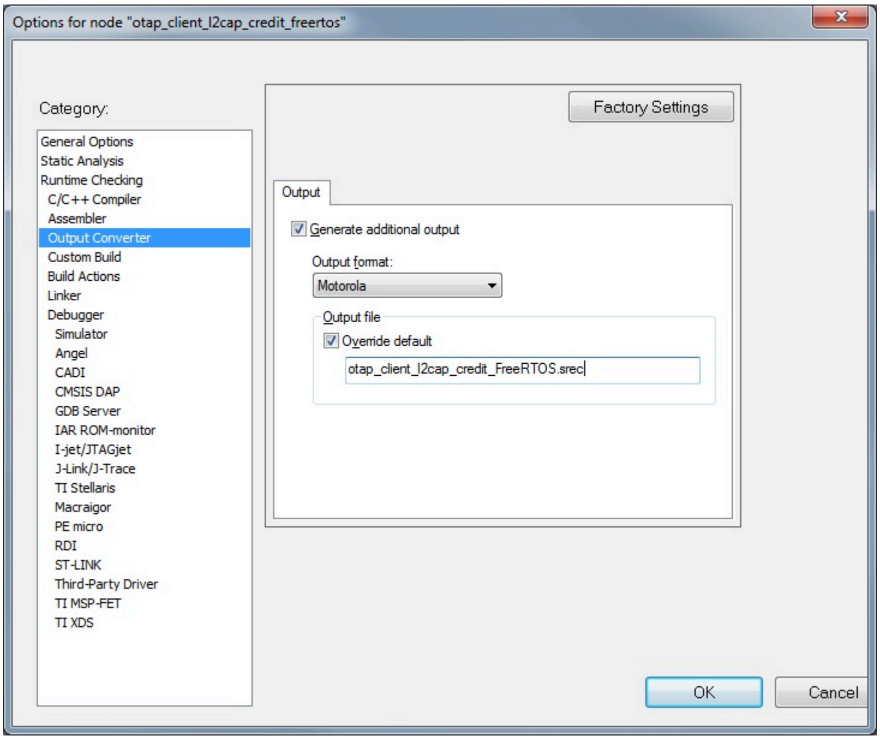


Figure 33. IAR Output Converter Dialog - .srec output

3. To obtain a `.bin` file from IAR, select the **Raw binary** option in the **IAR Output Converter** as seen in the Figure 34.

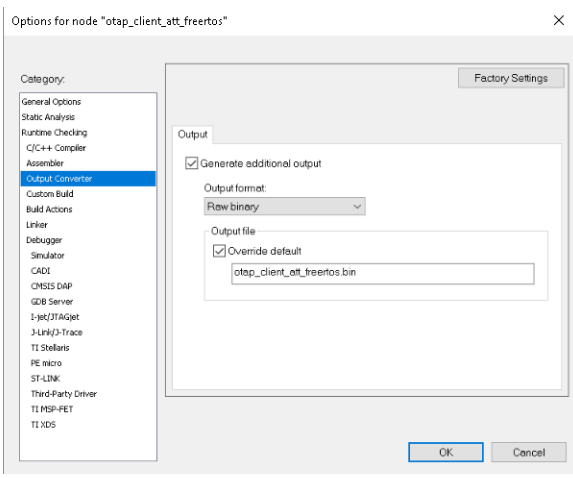


Figure 34. IAR .bin file output converter

4. 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 shows up. In this window, add the following command:

```
arm-none-eabi-objcopy -v -O binary  --only-section=.text
  --only-section=.data
 --only-section=.ARM.exidx "${BuildArtifactFileName}"
 "${BuildArtifactFileBaseName}.bin"
```

In case the command already exists, uncomment it by removing the '#' character at the beginning.

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  --only-section=.text
  --only-section=.data --only-section=.ARM.exidx"
 ${BuildArtifactFileName}" "${BuildArtifactFileBaseName}.s19"
```

This window is shown in Figure 35.

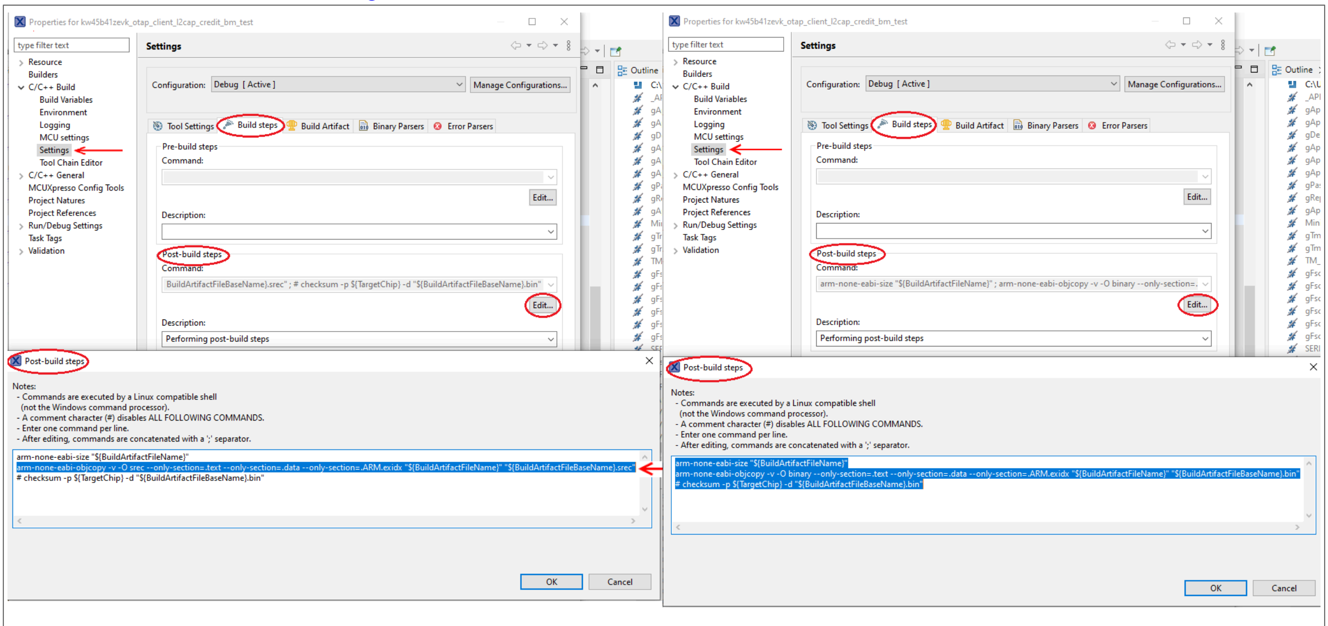


Figure 35. MCUx .bin .srec file output

5. Start the **Over The Air Programming** application and select "**OTAP Bluetooth LE**" from the "**Select OTA Protocol**" combo box as shown in Figure 36.

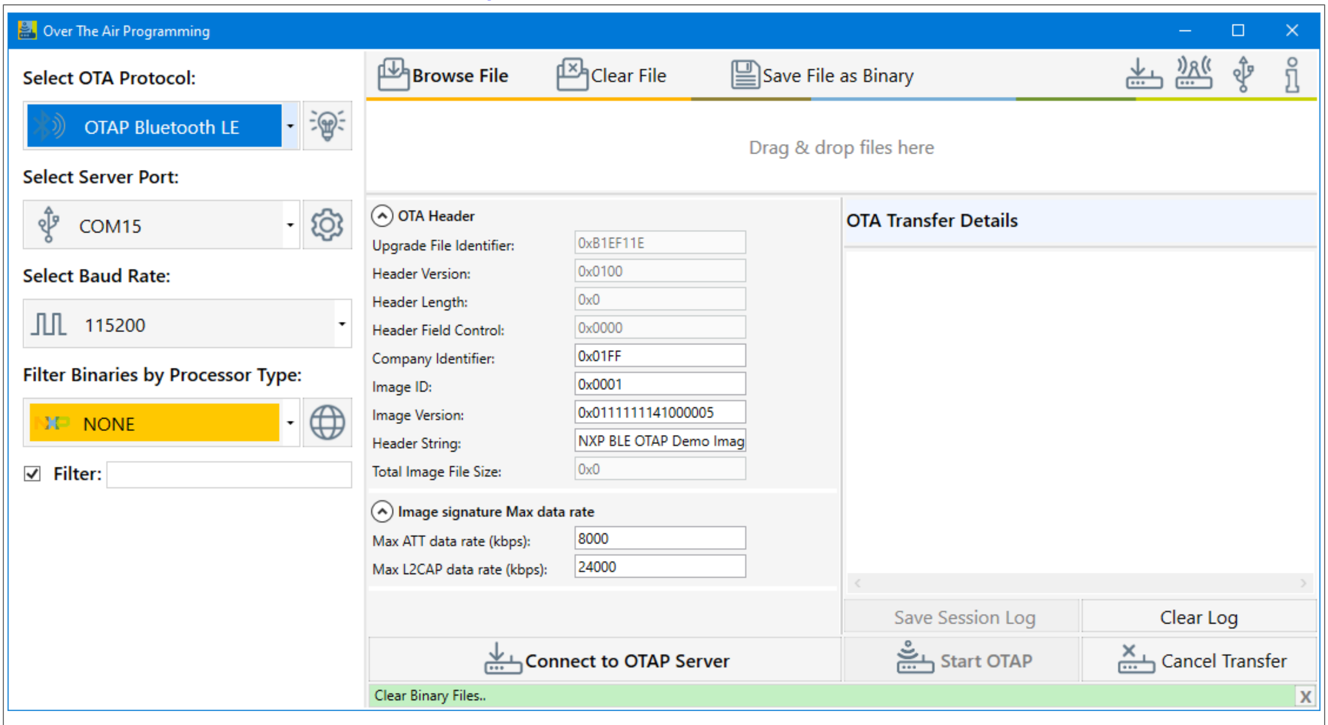


Figure 36. Test Tool - Selecting the OTAP Bluetooth LE protocol

6. **Load the image file into the application, then configure the image file header and start the OTAP Server:**
   - To select the updated image In the Over the Air Programming tool, select the "**Browse File**" button and then navigate to the `.srec` or `.bin` file containing the image to be sent to the OTAP Client. After the `.srec` or `.bin` file is chosen, a pop-up window asks to choose the target processor. Choose the **KW45/K32W** processor and press **OK**. See Figure 37.

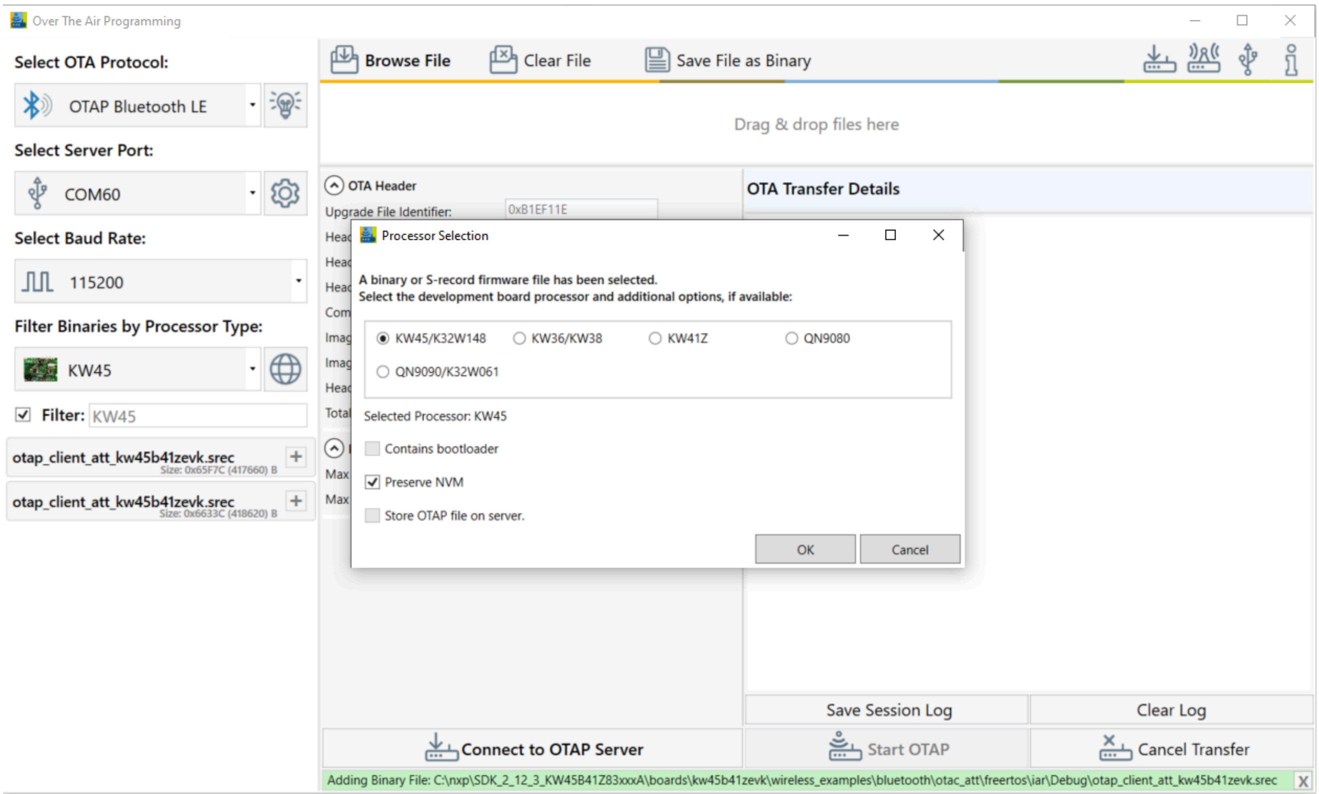


Figure 37. Over the Air Programming tool Bluetooth LE Processor Selection

   - Once the processor is selected, a new pop-up window would appear that allows selecting the type of image (KW45Z/K32W1(MCU)) as shown in the Figure 38. (In the specified case, we selected the KW45Z/KW45Z/K32W1(MCU)).

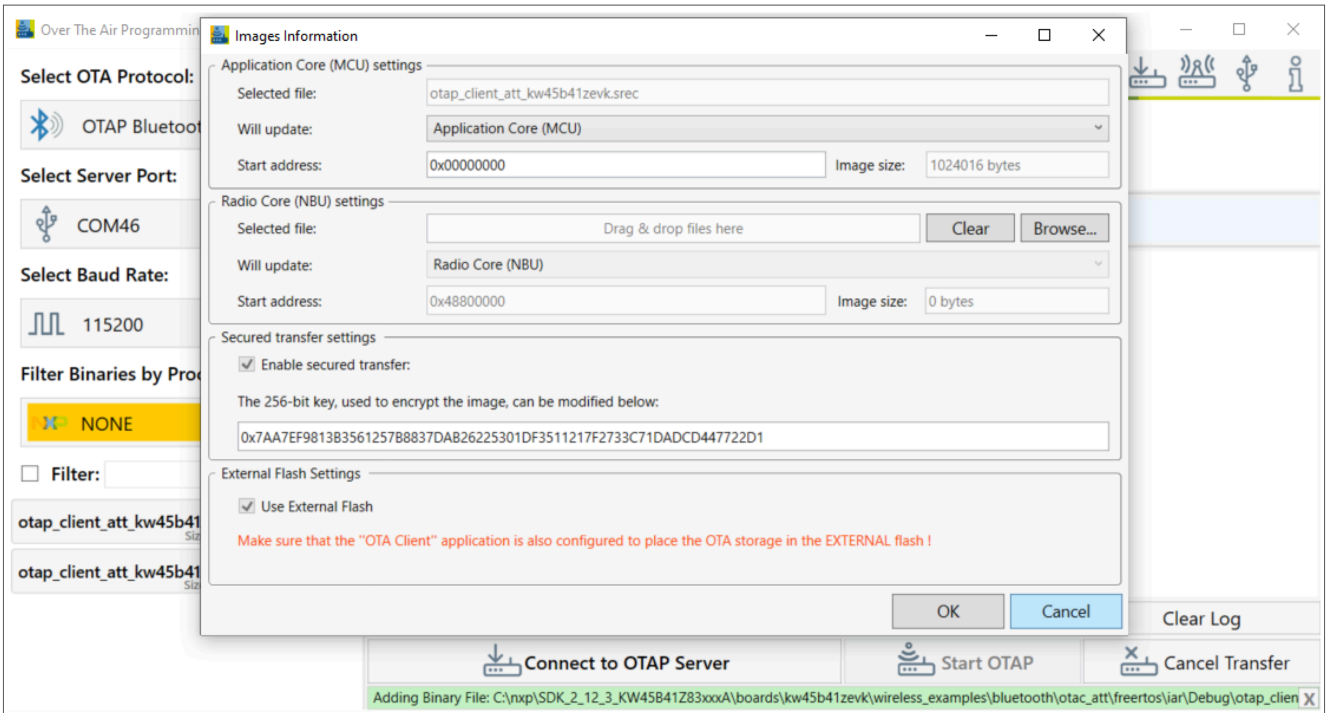


Figure 38. Selecting uploaded image type

- To update the KW45B/KW32W1 (radio) image, select it by pressing the "**Browse**" button in the M3 group. Then navigate to the `.bin` file as in the Figure 39.

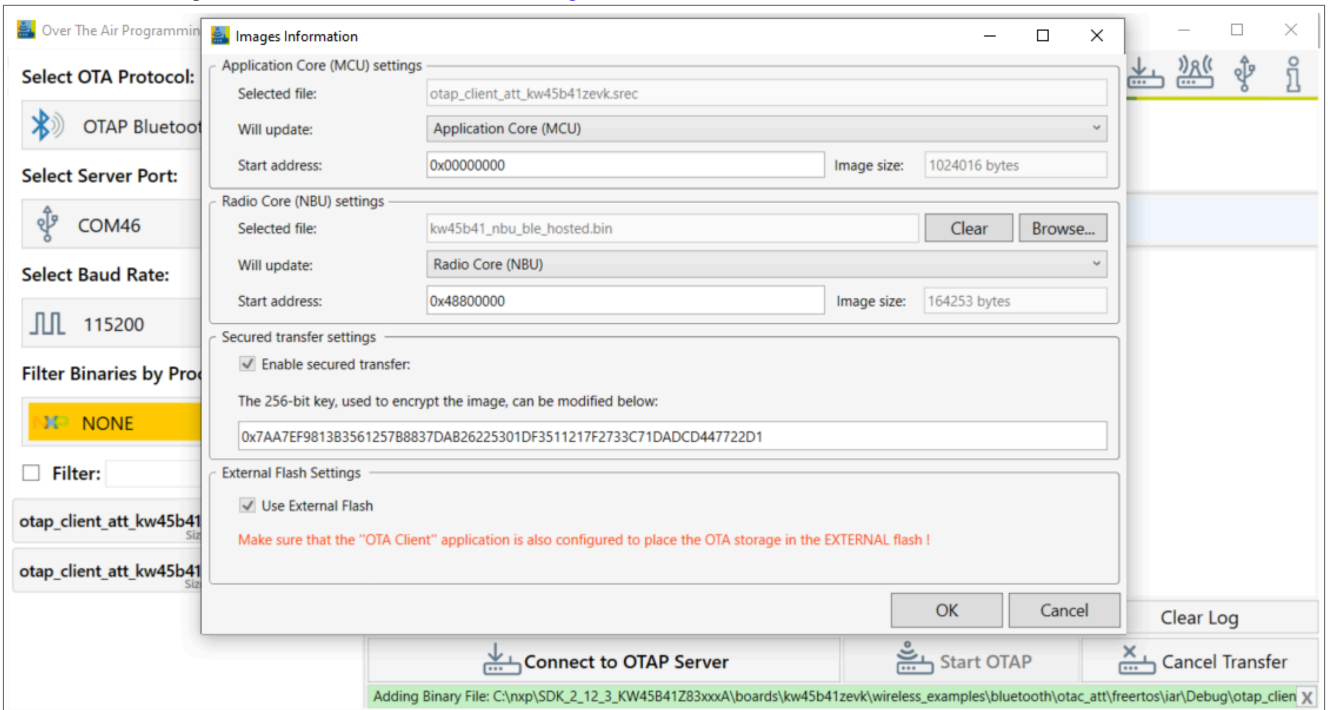


Figure 39. Selecting KW45B radio image

- If the OTA client has configured external memory support, then "**Use External Flash**" checkbox must be checked as in the figure below. If the OTA client has configured internal memory support, the checkbox must be left unchecked.

This checkbox (if checked) instructs the OTA client bootloader to erase all the internal storage. This must be done only if external memory support is used as shown in Figure 40.

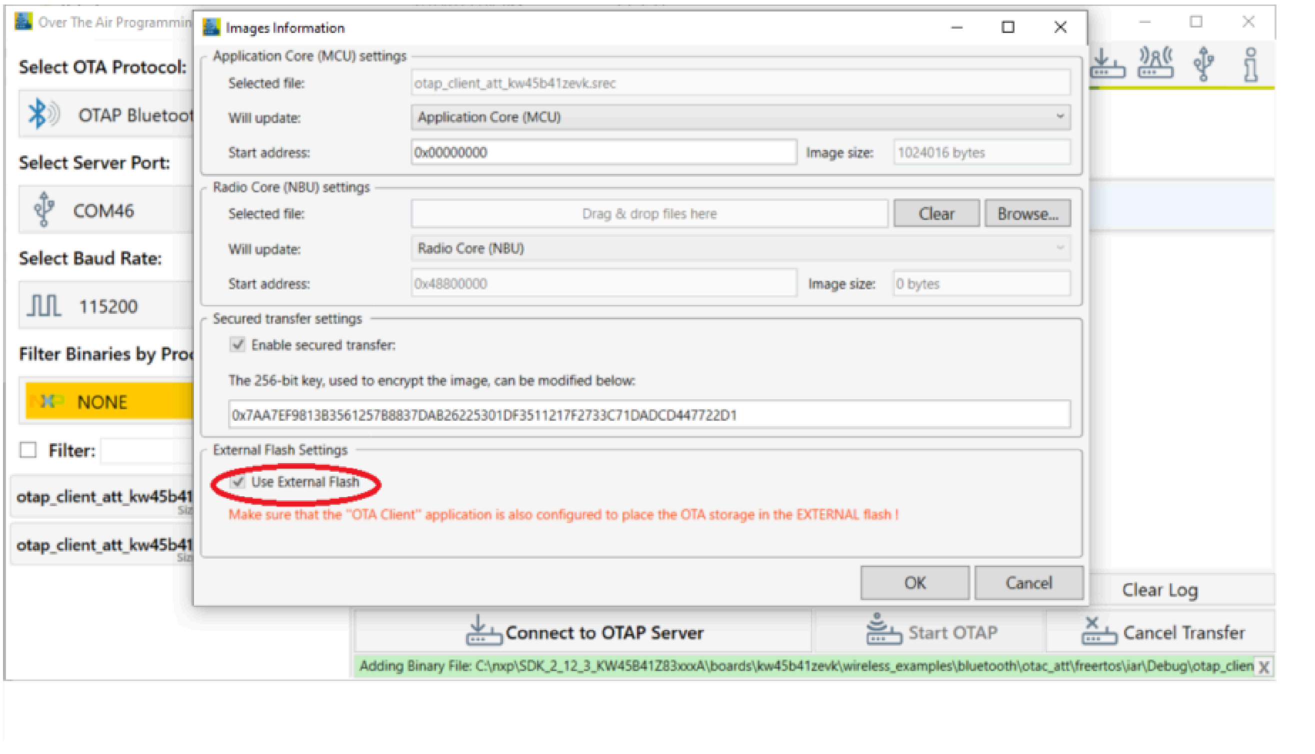


Figure 40. 'Use External Flash' checkbox

At this moment, click the **OK** button. A new window appears that prompts you to enter a location where the secured file should be stored. By default, the location of the original file is selected.

Note: *The extension of the secured file is *.sb3. See Figure 41.*

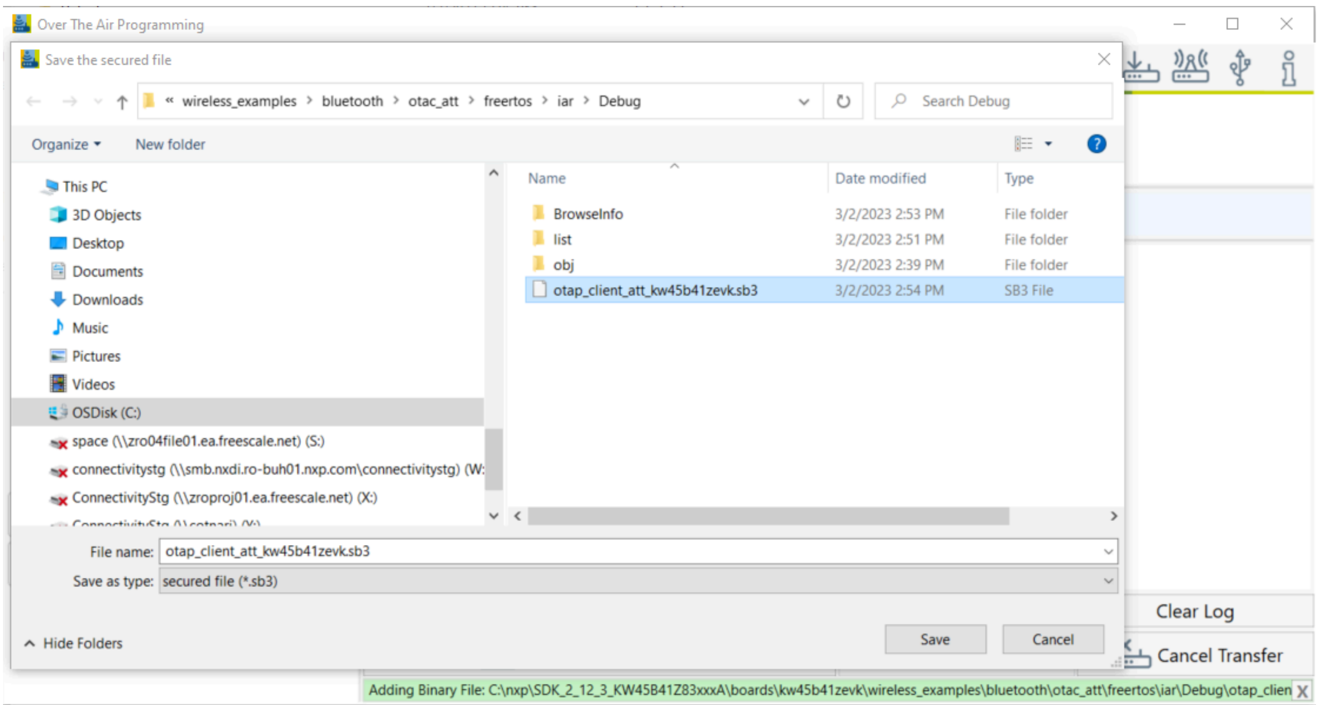


Figure 41. Selecting location to save *.sb3 file

- You can now configure two different JSON files, used to:
  - Sign the file that is uploaded to the MCU if an MCU file was selected.
  - Create the **.sb3* container that is sent OTA. The **.sb3* file can contain only the MCU file, only the radio file, or both.

  If you select a file that is written on the MCU, a new window appears as shown in the figure below. This window helps in configuring the root certificates and signing the certificates, by either dragging and dropping, or browsing for new files. For details on each field of the JSON, see */Documentation/KW45Json Description.pdf* provided with Over the Air Programming tool.

  By default, the JSON is configured for the demo applications to run as shown in Figure 42.

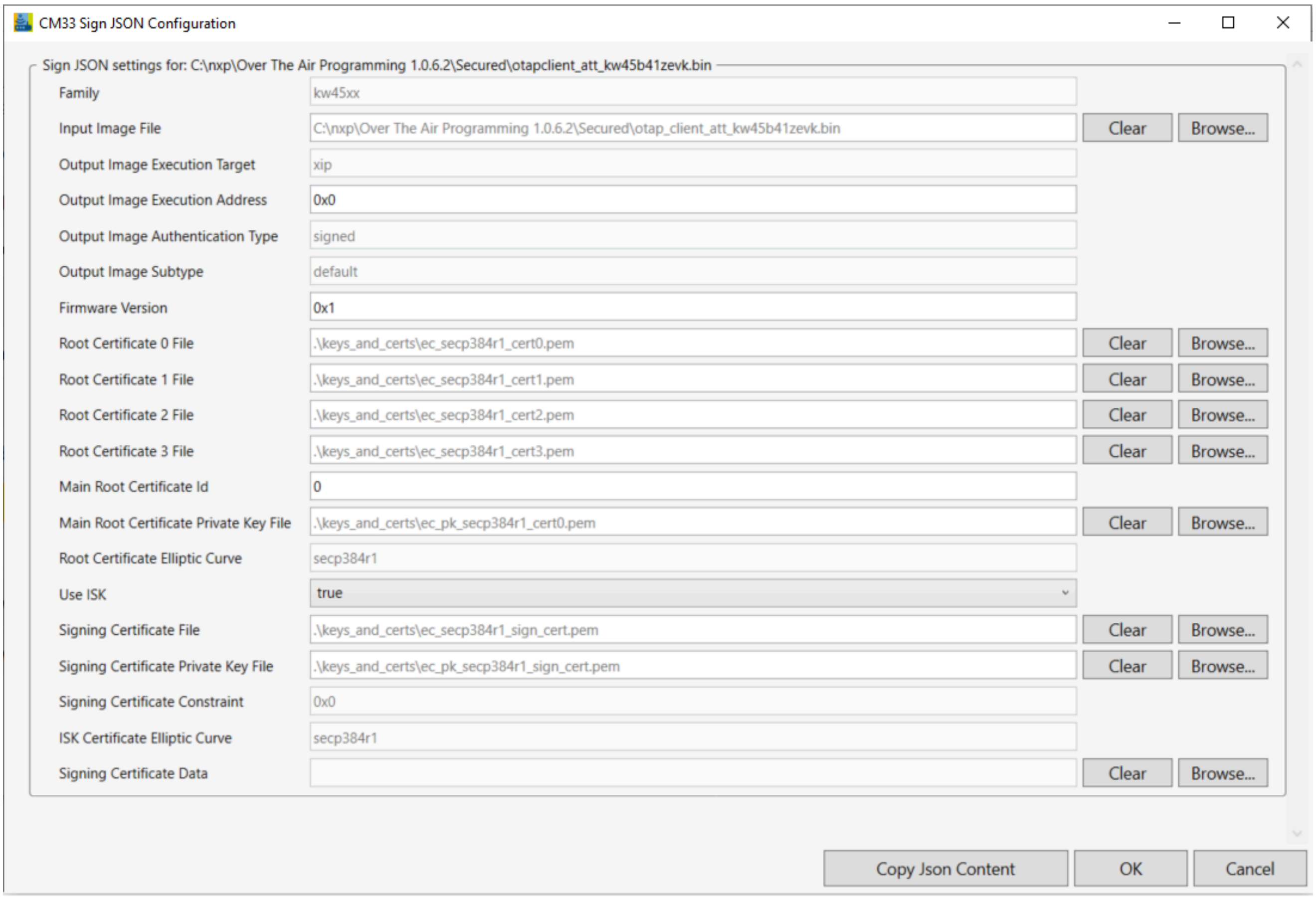


Figure 42. CM33 sign JSON configuration

After configuring the JSON file used for signing the MCU file, a new similar window appears. As shown in the Figure 43, the window is designed for configuring the **.sb3* container. This window helps you to configure the encryption key file, the root certificates, and the signing certificates by either drag and dropping or browsing for new files. For details on each field of the JSON file, see */Documentation/KW45 JsonDescription.pdf* provided with Over the Air Programming tool.

By default, the JSON is configured for the demo applications to run as shown in Figure 43.

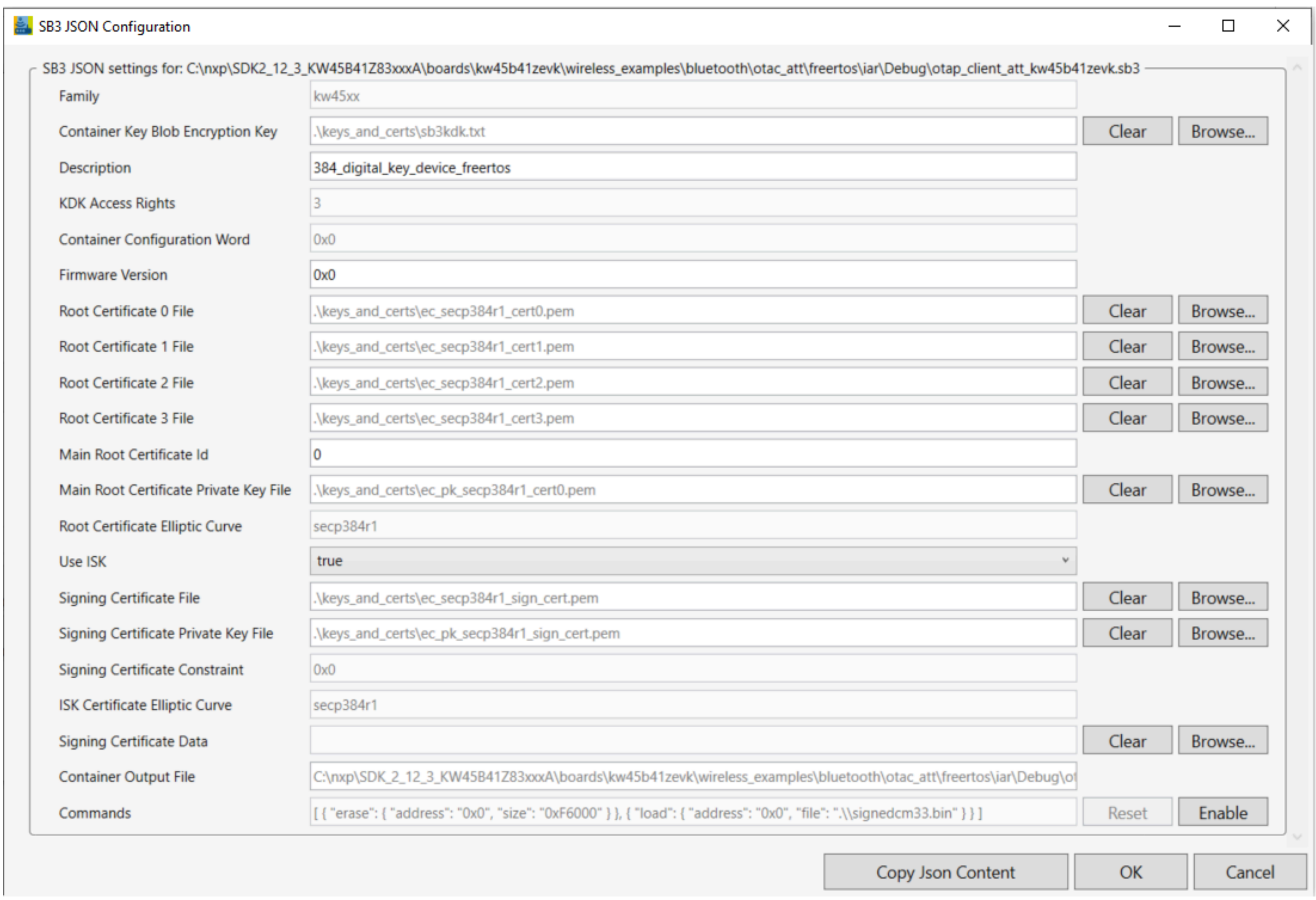


Figure 43. SB3 JSON configuration

- After the `.sb3` file is created, the "**Encryption Key**" and "**Authentication Key**" are presented. For the secured update to be successful, the destination board must have been provisioned with these keys through fuse burning, as described in the accompanying document. Depending on the board type, it can either be already provisioned by NXP (KW45B41Z-EVK / K32W148-EVK samples) or not provisioned (loosen samples). See Figure 44

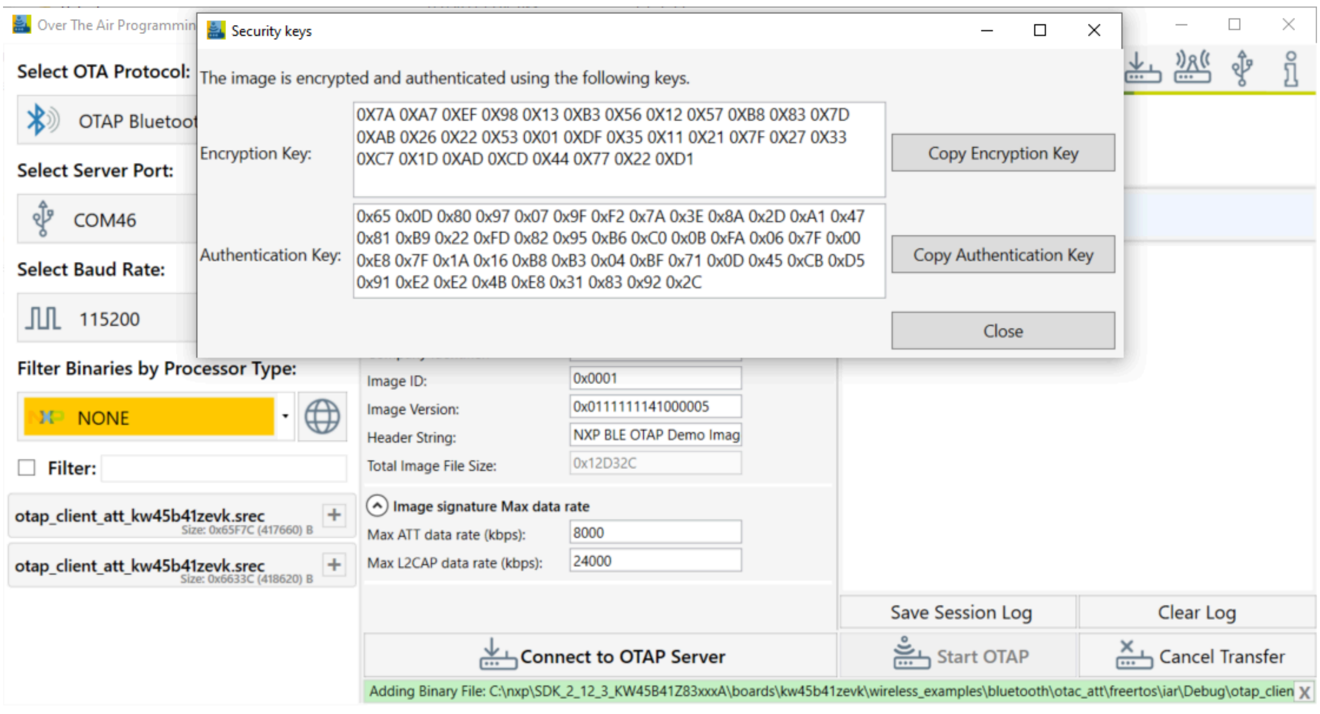


Figure 44. Encryption key and authentication key

- The OTA Header configuration options from the "**OTA Header**" box are 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 image is loaded, go to the "**Select Server Port**" box, select the correct COM Port for the OTAP Server board. Also select the default baud rate of 115200 and press the "**Connect to OTAP Server**" button. A successful connection is displayed in the Message Log.
  - If the image is loaded before connecting to the OATP Server COM Port, then the OTAP Server of the application starts automatically.
  - If the connection to the COM Port is established before the image is loaded, then the "**Start OTAP**" button must be pressed to start the OTAP Server of the application. For details, see the Figure 45 below.

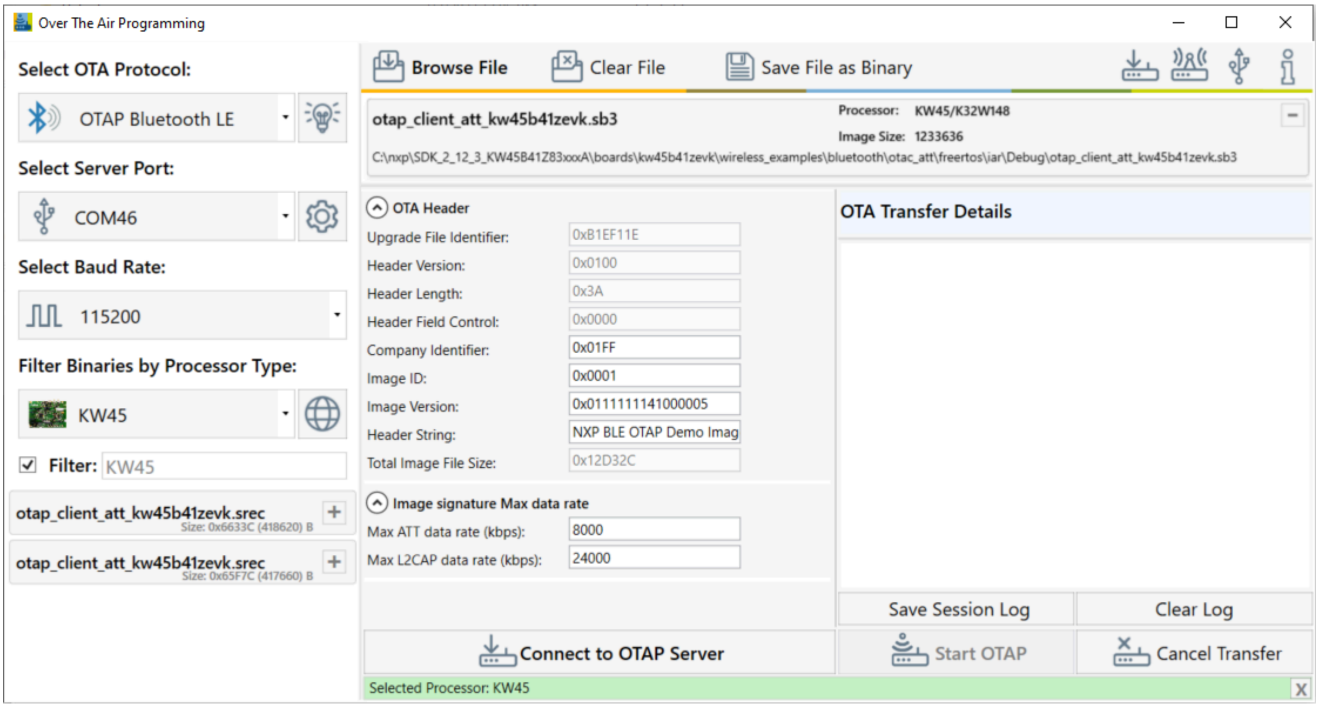


Figure 45. Over the Air Programming application overview

- 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. Data rate can significantly slow down if configuration is not done correctly and errors appear in the transfer process.
  The optimal data rate depends on multiple factors. Some of these factors are listed below:
  – 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 the OTAP Server board
  If the data rate is too high, then the OTAP Client receives a new chunk before it can process the previous one. In such a case, it sends an "**Unexpected Chunk Sequence Number**" error and restarts 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 might not reach the client triggering a similar type of error. The default data rate values should work for most configurations.

7. 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. See Figure 46.

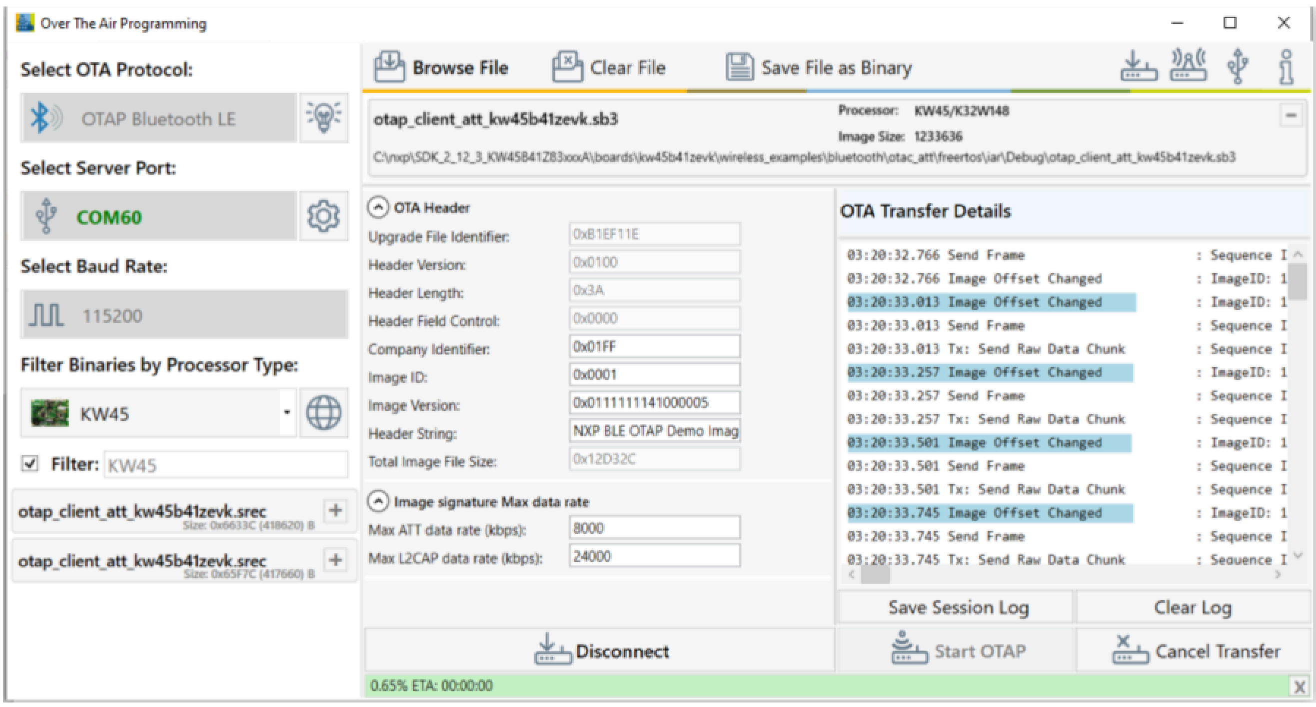


Figure 46. Test Tool OTAP Bluetooth LE image transfer in progress

8. After all the blocks are sent, the OTAP Client sends an `Image Transfer Complete` command to the OTAP Server. When the PC Application receives this command, it displays a `Sent Image with Success` message in the log window. See Figure 47.

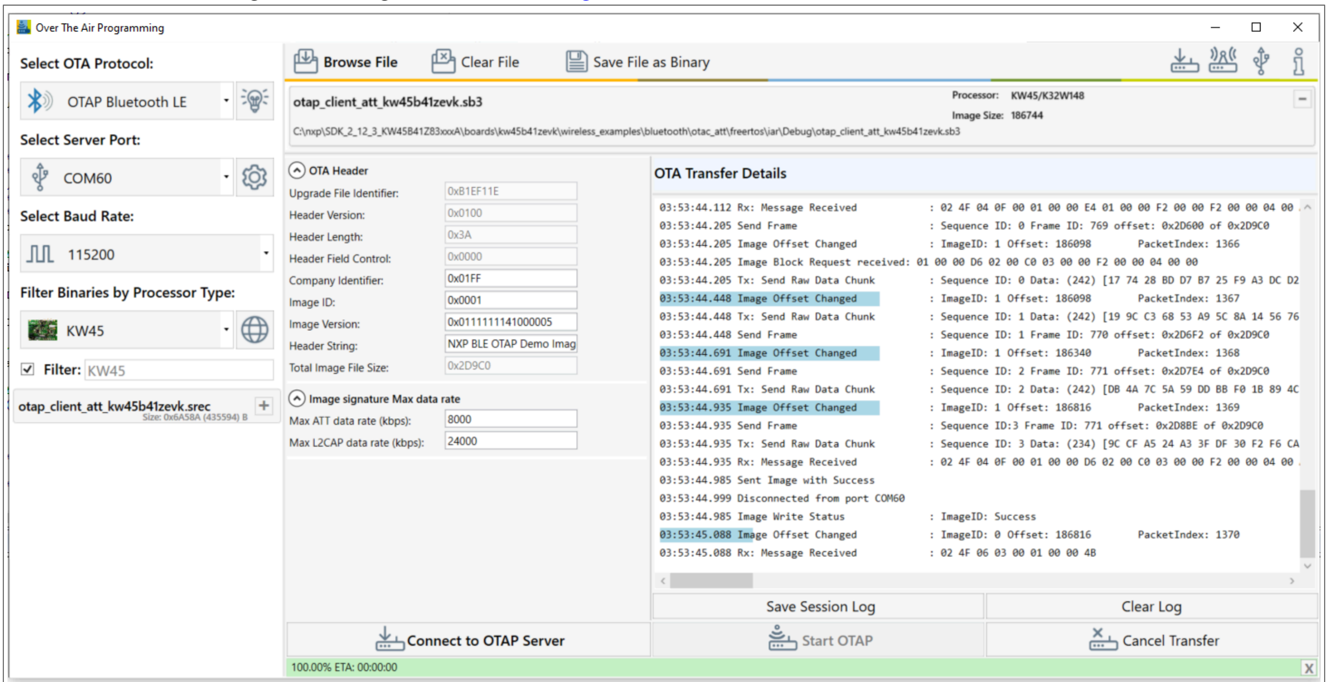


Figure 47. Test Tool OTAP Bluetooth LE Image Transfer Completed

9. 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 frame, the MCU resets again and runs the new image.

5.11.5 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, perform the following steps:

1. Flash the OTAP Client ATT to either the KW45B41Z-EVK or the K32W148-EVK platform. The Kinetis Bluetooth LE Toolbox only supports the ATT OTAP Client.
2. In order to send over the air in `.bleota` format, create the application. In order to load the image file into the Over the Air Programming application and create the .sb3 file, follow the instructions described in Section 5.11.4 "Usage with Over The Air Programming Tool". See Figure 48 Once the `.sb3` file is created, press the "**Save File as Binary**" button to create the `.bleota` file.

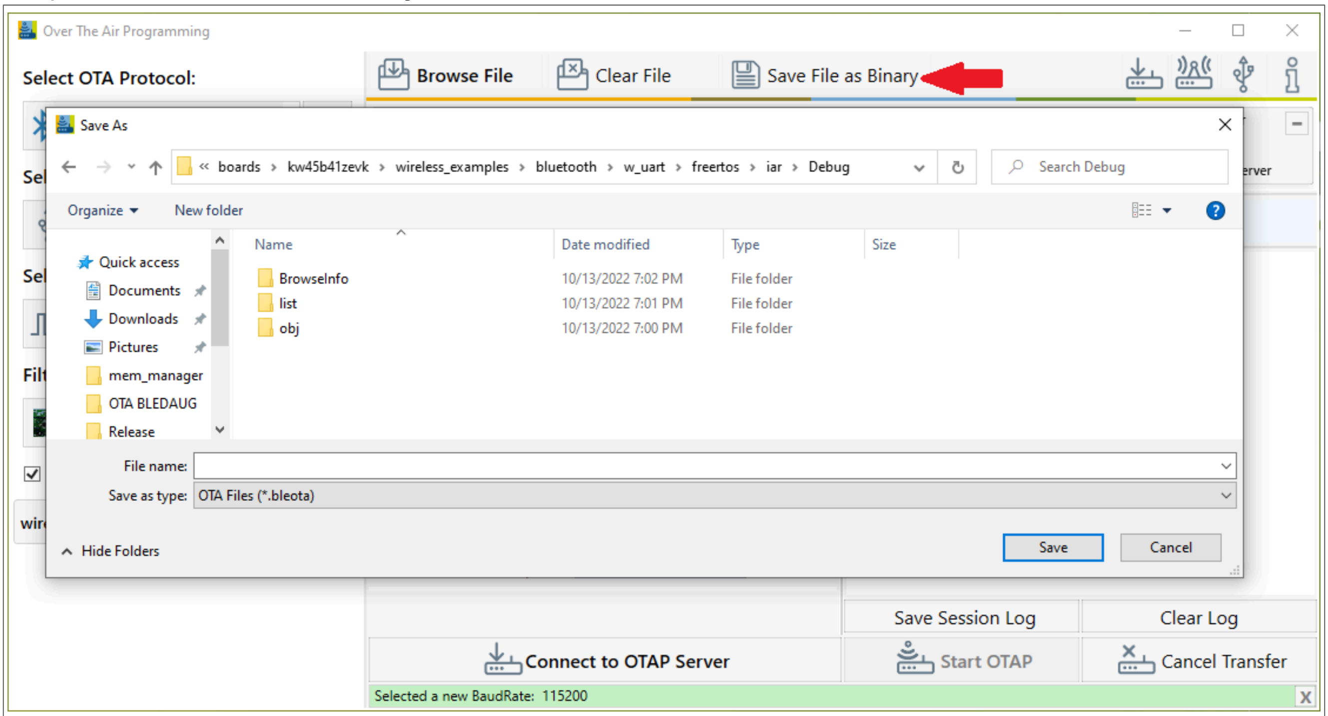


Figure 48. Save file in `*.bleota` file format

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 Figure 49.

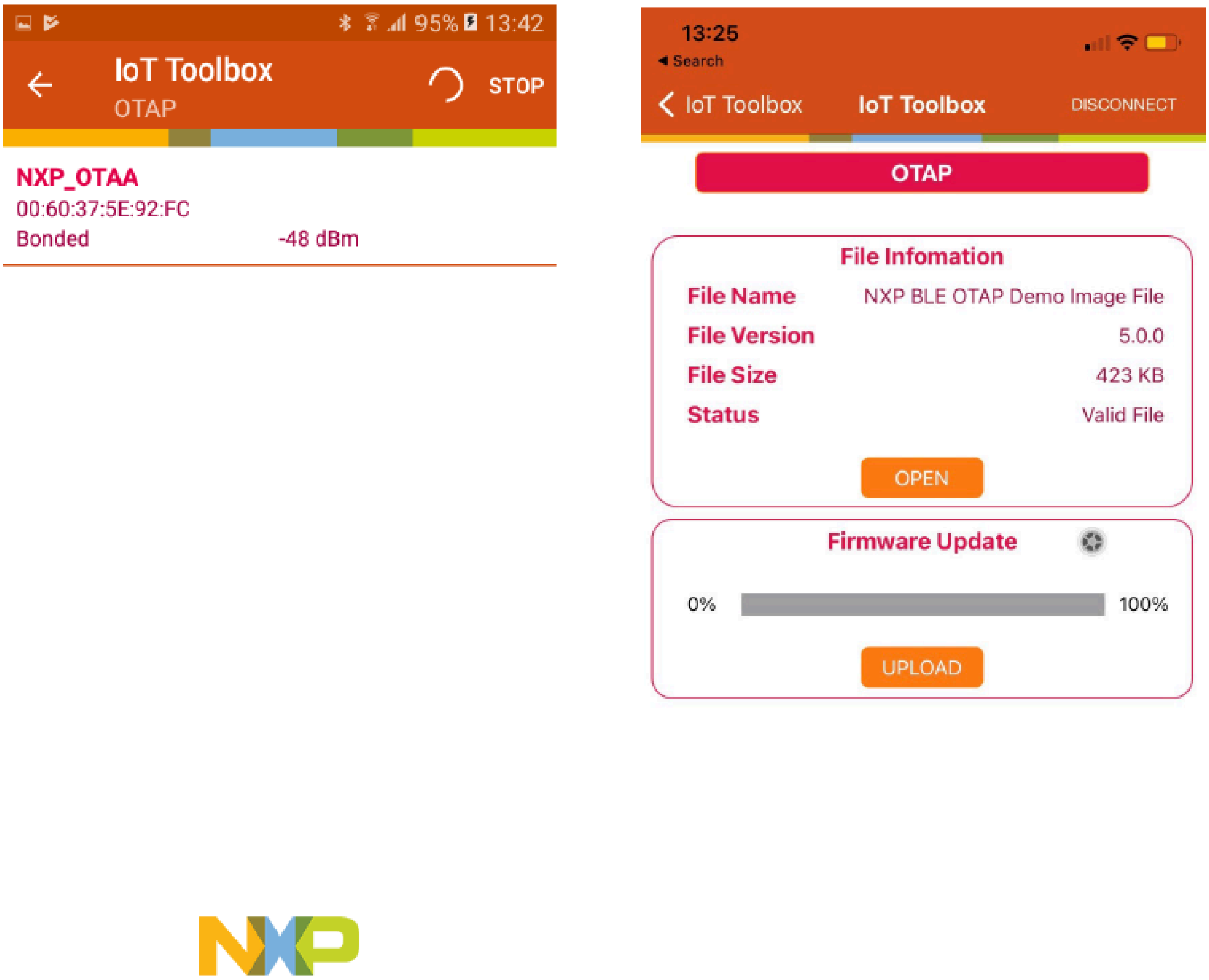


Figure 49. IoT Toolbox - OTAP Tool Scanning and Discovery

5. Press the "**Open**" button and load the `.bleota` 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 progress bar displays 100% update after a successful transfer, as shown in Figure 50.

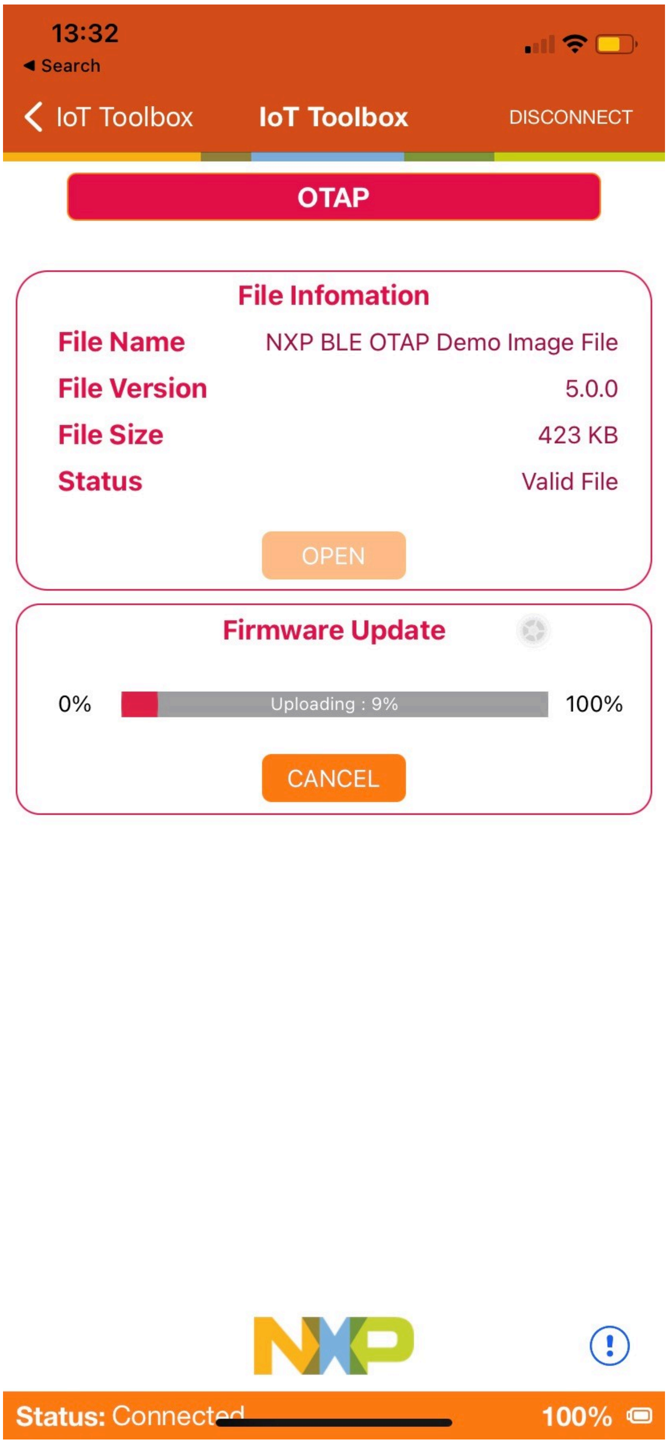


Figure 50. IoT Toolbox - OTAP Image Transfer

6. 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.

5.12 Wireless UART

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

5.12.1 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.

5.12.2 Supported platforms

The following platforms support the Wireless UART application:

- KW45B41Z-EVK
- K32W148-EVK
- FRDM-MCXW71
- KW47-EVK
- MCXW72-EVK
- FRDM-MCXW72

5.12.3 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 of 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 Table 11 below for hardware references.

Table 11. Hardware references

Platform	SCANSW	CONNLED	ROLESW
KW45B41Z-EVK / K32W148-EVK	SW2	LED2	SW3
FRDM-MCXW71	SW2	Blue LED	SW4
KW47-EVK / MCXW72-EVK	SW2	LED2	SW3
FRDM-MCXW72	SW4	Blue LED	SW2

5.12.4 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, perform the following steps:

1. Open a serial port terminal and connect them to the two boards, in the same manner described in Section 4.3 "Testing devices". The start screen is blank after the board is reset.
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.
4. After creating a connection, the role (central or peripheral) is displayed on the console. The role switch can be pressed again before creating a new connection.

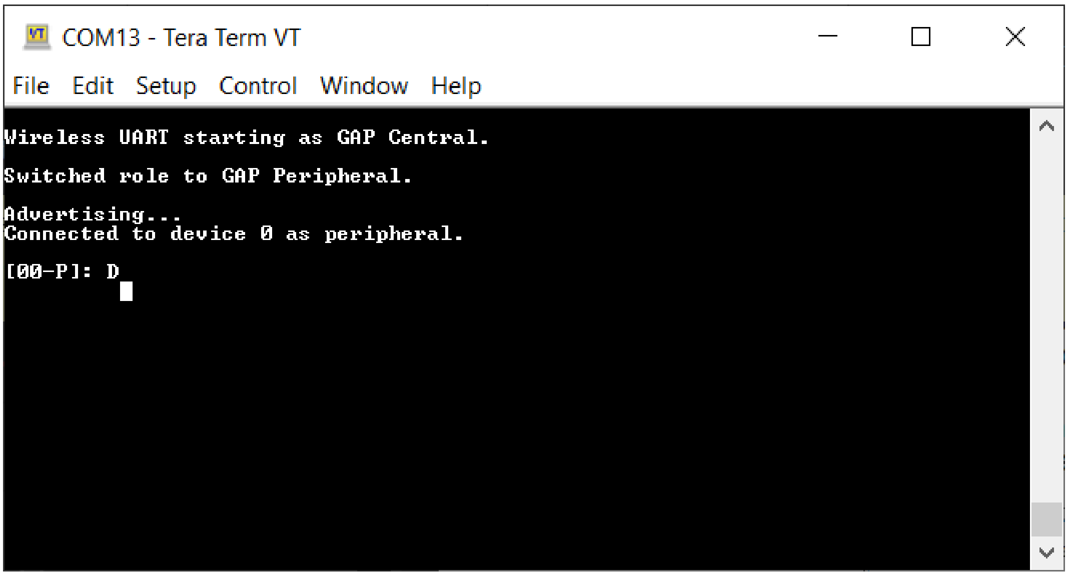


Figure 51. Tera Term – received text on Wireless UART

When testing with a single board and the IoT Toolbox, perform the following steps:

1. Open a serial port terminal and connect the board in the same manner described in Section 4.3 "Testing devices". 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** button to start advertising. The IoT Toolbox app can then connect. Select UART instead of Console and start typing, as shown in the Figure 51.

5.13 Bluetooth LE Shell

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

5.13.1 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.

5.13.2 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)

5.13.3 Supported platforms

The following platforms support the Bluetooth LE Shell application:

- KW45B41Z-EVK
- K32W148-EVK
- FRDM-MCXW71
- KW47-EVK
- MCXW72-EVK
- FRDM-MCXW72

5.13.4 User interface

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

5.13.5 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 4.3 "Testing devices". 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.

```
BLE Shell>gap devicename HRS
--> GATTDB Event: Attribute Written
HRS>gap advdata 1 6
HRS>gap advdata 8 HRS
HRS>gap advstart
HRS>
```

 All information provided in this document is subject to legal disclaimers.

```
--> GAP Event: Advertising parameters successfully set.
HRS>
--> GAP Event: Advertising data successfully set.
HRS>
--> GAP Event: Advertising state changed successfully!
HRS>gattdb addservice 0x180D
--> Heart Rate
- Heart Rate Measurement Value Handle: 14
```

3. Configure the other device as a GAP central. Change its name to 'Collector'. Start scanning and connect to the HRS device by selecting the corresponding device index from the list of scanned devices. In the example below, the HRS device is device number 2. The number of listed scanned devices can be controller through the `mShellGapMaxScannedDevicesCount_c` define in `shell_gap.c`.

```
nBLE Shell>gap devicename Collector
--> GATTDB Event: Attribute Written
Collector>gap scanstart filter
--> 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 (gAppUsePairing_d and gAppUseBonding_d must be set in app_preinclude.h). On the collector initiate the pairing.

```
Collector>gap pair 0
--> Pairing...
--> GAP Event: Link Encrypted
--> GAP Event: Device Paired
```

5. 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 Attribute Start Handle: 1 End Handle: 4
- Service Changed Value Handle: 3
- Client Characteristic Configuration Descriptor Handle: 4
--> Generic Access Start Handle: 5 End Handle: 11
- Device Name Value Handle: 7
- Appearance Value Handle: 9
- Peripheral Preferred Connection Parameters Value 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
```

6. 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
--> GATT Event: Characteristic Value Written!
```

7. 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
```

8. A notification appears on the Collector console.

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

5.13.5.1 Extended advertising

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

On the GAP Peripheral device:

1. 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.
2. Configure the extended advertising data. The Bluetooth LE Shell applications has the feature to send for test, a large data payload. Use the extended advertisement default configuration (not call "`gap extadvcfg`"), pass the command "`gap extadvdata`" with no parameters 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`. Start the default test with call for "`gap extadvstart`".
   The advertising data type is set to shortened local name (8) and the advertising data content is set to `test_ext_adv_data`.
   Note: *Users must note that extended connectable advertising does not allow for chained advertising data. The data length must be limited to what can fit in a single AUX_ADV_IND PDU (251 bytes at maximum). This means that passing the `gap extadvdata` with no parameters and the default value of `SHELL_EXT_ADV_DATA_SIZE` (500 bytes) after having set the advertising type to connectable will result in an error when trying to start advertising.*
3. Start extended advertising.

```
BLE Shell>gap extadvcfg -type 65 -phy1 3
BLE Shell>gap extadvdata 8 test_ext_adv_data
BLE Shell>gap extadvstart
--> GAP Event: Extended
Advertising parameters successfully set.
--> GAP Event:
Extended Advertising data successfully set.
--> GAP Event: Advertising state changed successfully!
```

4. **On the GAP Central device**
   Set the scanning parameters. The scanning PHY is set to match the advertising PHY, in this case Coded PHY.
5. Start scanning and filter duplicates.

```
BLE Shell>gap scancfg -phy 4
BLE Shell>gap scanstart filter
BLE Shell>
-> GAP Event: Scan started.
BLE Shell>
--> GAP Event: Found device 0 : 0060375BCEC6 -23 dBm
Advertising Extended Data:
test_ext_adv_data
```

6. 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
BLE Shell>
-> GAP Event: Scan stopped.
BLE Shell>
--> GAP Event: Connected to peer 0
```

5.13.5.2 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.

1. **On GAP peripheral device**
   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.

```
BLE Shell>gap address
BLE Shell>
--> GAP Event: Public Address Read:C4603770BCC5
BLE Shell>gap extadvcfg -phy1 3
BLE Shell>gap extadvstart
BLE Shell>
--> GAP Event: Extended Advertising parameters successfully set.
BLE Shell>
--> GAP Event: Extended Advertising data successfully set.
BLE Shell>
--> GAP Event: Advertising state changed successfully!
BLE Shell>gap txpower adv 0
BLE Shell>
--> GAP Event: Success!
```

2. **On GAP Central device**
   Set the scanning PHY to Coded PHY. Start monitoring the RSSI on advertising Channel using the address of the Peripheral device. Scanning starts automatically, if it is not previously enabled.

```
BLE Shell>gap scancfg -phy 4
BLE Shell>gap rssimonitor C4603770BCC5--> Reading RSSI on advertising
 channel:
BLE Shell>
-> GAP Event: Scan started.
BLE Shell>
 RSSI: -27 dBm
 RSSI: -27 dBm
 RSSI: -27 dBm
 RSSI: -27 dBm
 RSSI: -27 dBm
 RSSI: -29 dBm
```

In the below example, the RSSI in monitored 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
BLE Shell>gap extadvdata 8 rssimonitortest
BLE Shell>gap extadvstart
BLE Shell>
--> GAP Event: Extended Advertising parameters successfully set.
BLE Shell>
--> GAP Event: Extended Advertising data successfully set.
BLE Shell>
--> 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
BLE Shell>
--> GAP Event: Success!
```

On the GAP Central device, start scanning on the Coded PHY. Update the connection PHY also to Coded PHY, then connect to the 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
BLE Shell>
->  GAP Event: Scan started.
BLE Shell>
-->  GAP Event: Found device 0 : C4603770BCC5 -21 dBm
 Advertising Extended Data:
rssimonitortest
gap connect 0
BLE Shell>
->  GAP Event: Scan stopped.
BLE Shell>
-->  GAP Event: Connected to peer 0
BLE Shell>gap rssimonitor 0 -c
-->  Reading RSSI from connected device:
BLE Shell>
 RSSI: -22 dBm
 RSSI: -23 dBm
 RSSI: -21 dBm
 RSSI: -22 dBm
 RSSI: -22 dBm
BLE Shell>gap rssistop
```

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

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

```
BLE Shell>gap phy 0 -tx 2 -rx 2
BLE Shell>
--> GAP Event: Phy update complete with peer 0
--> TxPhy: 2M
--> RxPhy: 2M

BLE Shell>gap rssimonitor 0 -c
--> Reading RSSI from connected device:
BLE Shell>
RSSI: -23 dBm
RSSI: -23 dBm
RSSI: -21 dBm
RSSI: -21 dBm
--> GAP Event: Phy update complete with peer 0
--> TxPhy: Coded
--> RxPhy: Coded

BLE Shell>
RSSI: -22 dBm
RSSI: -21 dBm
RSSI: -22 dBm
RSSI: -23 dBm
RSSI: -23 dBm
--> GAP Event: Phy update complete with peer 0
--> TxPhy: 2M
--> RxPhy: 2M
BLE Shell>
RSSI: -22 dBm
RSSI: -21 dBm
RSSI: -21 dBm
RSSI: -20 dBm
```

5.13.6 Throughput feature

The Bluetooth LE Shell application also has a throughput test feature that can be used to test different combinations of the parameters (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 displays a report

All throughput-related commands are grouped under the **`thrput`** keyword:

- `thrput setparam`: configures connection interval, packet count and payload size.
- `thrput start tx`: configures the device as a GAP Peripheral and starts advertising. Once the receiving device is connected, the packet transmission begins. 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 starts scanning. Once a transmitter device is found, it connects to it and waits for the test to begin. The connection interval can also be configured (`-ci` <value>).
- `thrput stop`: stops the test and disconnects the devices.

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

The receiving device generates the report if no packets are received for more than three consecutive connection events.

The default configuration of the throughput test is the following:

- Packet count: 1000
- Payload size: 20 bytes

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

The example of a test report is shown below:

```
BLE Shell>thrput start tx
BLE Shell>
-->  GAP Event: Advertising parameters successfully set.
BLE Shell>
-->  GAP Event: Advertising data successfully set.
BLE Shell>
-->  GAP Event: Advertising started.
-->  GAP Event: Connected to peer 0
BLE Shell>
-->  GAP Event: Advertising Throughput test started.
Sending packets...
-->  MTU Exchanged.
BLE Shell>
Throughput test with peer 0 has finished.

BLE Shell>thrput start rx
BLE Shell>
->  GAP Event: Scan started.
Found device:
THR_PER
0060375BCEC6
->  GAP Event: Scan stopped.
-->  GAP Event: Connected to peer 0
BLE Shell>Throughput test started.
Receiving packets...


************************************
***** TEST REPORT FOR PEER ID 0 ****
************************************
Packets received: 1000
Total bytes: 244000
Receive duration: 5017 ms
Average bitrate: 389 kbps
************************************
********** END OF REPORT ***********
```

5.13.7 Decision-Based Advertising Filtering (DBAF) feature

The Bluetooth LE Shell application also supports the Decision-Based Advertising Filtering (DBAF) feature, which can be enabled by performing the following steps:

- Flash the experimental NBU found in `middleware/wireless/ble_controller/bin/experimental`.
- Update the application project so that it uses the experimental Bluetooth LE Host library: `middleware/wireless/bluetooth/host/lib_exp/lib_ble_OPT_host_cm33_iar.a`
- In `app_preinclude.h` file, set `BLE_SHELL_DBAF_SUPPORT` to `1`.

This feature requires two devices:

- GAP Peripheral: transmits decision PDUs (ADV_DECISION_IND).
- GAP Central: scans for decision PDUs and handles filtering policies.

To showcase the functionality, two platforms are used in the following setup.

Steps to perform on the GAP Peripheral device:

1. Configure the extended advertising parameters to use decision PDUs. In the below example, the advertising type is set to connectable and includes TX power, uses decision PDUs and includes AdvA in the decision PDU. The primary PHY is set to Coded PHY.
2. Configure the extended advertising data using the `gap extadvdecdata` command. The resolvable tag and/or arbitrary data can be set using the parameters available.
3. Start extended advertising.

```
BLE Shell>gap extadvcfg -phy1 3 -type 449
BLE Shell>gap extadvdecdata -key 112233445566778899AABBCCDDEEFF00 -prand
 5AC317 -decdata 6362 -datalen 2 -restag 0
BLE Shell>gap extadvstart
BLE Shell>
--> GAP Event: Extended Advertising parameters successfully set.
BLE Shell>
--> GAP Event: Extended Advertising data successfully set.
BLE Shell>
--> GAP Event: Extended Advertising Decision Data Setup Complete.
BLE Shell>
--> GAP Event: Advertising state changed successfully!
BLE Shell>
```

Steps to perform on the GAP Central device:

1. Set the scanning parameters to scan only decision PDUs. The scanning PHY is set to match the advertising PHY, in this case Coded PHY.
2. Set the connection parameters to use only decision PDUs and the connection initiating PHYs corresponding to the primary PHY on which the advertising is performed.

```
BLE Shell>gap scancfg -phy 4 -filter 12
BLE Shell>gap connectcfg -phy 4 -filter 2
--> Connection Parameters:
 --> Connection Interval: 200 ms
 --> Connection Latency: 0
 --> Supervision Timeout: 32000 ms
 --> Connecting PHYs: Coded
 --> Connection Filter Policy: 2
BLE Shell>
```

3. Add decision instructions using the `gap adddecinstr` command. A maximum of `gMaxNumDecisionInstructions_c` instructions can be added. If the set of instructions must be changed, the `gap deldecinstr` command deletes all current instructions.

```
BLE Shell>gap adddecinstr -group 1 -field 0 -criteria 1 -restagkey
 112233445566778899AABBCCDDEEFF00
BLE Shell>gap adddecinstr -group 1 -field 6 -criteria 1 -advmode 6
BLE Shell>gap adddecinstr -group 0 -field 24 -criteria 1 -arbmask
 00000000ffffff
BLE Shell>gap adddecinstr -group 0 -field 9 -criteria 1 -advacheck 2 -
add1type 0 -add1 a6fb0d376000 -add2type 0 -add2 a5fb0d376000
BLE Shell>gap adddecinstr -group 0 -field 7 -criteria 1 -rssimin -80 -rssimax
 0
BLE Shell>gap adddecinstr -group 0 -field 8 -criteria 5 -lossmin 0 -lossmax
 50
BLE Shell>
```

4. Set the decision instructions using the `gap setdecinstr` command. The instructions are used when listening for advertisements containing decision PDUs.
5. Start scanning and filter duplicates.
6. Connect to the desired device in the scanned devices list.

```
BLE Shell>gap setdecinstr
BLE Shell>
--> GAP Event: Decision Instructions Setup Complete.
BLE Shell>gap scanstart filter
BLE Shell>
-> GAP Event: Scan started.
BLE Shell>
--> GAP Event: Found device 0 : C4603770BCC5 -23 dBm
 Advertising Extended Data:
gap connect 0
BLE Shell>
-> GAP Event: Scan stopped.
BLE Shell>
--> GAP Event: Connected to peer 0
BLE Shell>
```

5.14 Hybrid (Dual-mode) Bluetooth Low Energy and Generic FSK

The Hybrid (Dual-mode) Bluetooth Low Energy and Generic FSK application demonstrates Generic FSK transmission/reception and Bluetooth advertising/scanning/multiple connections coexistence.

The Bluetooth LE part of this demo implements a modified version of the Wireless UART demo application, capable of multiple Bluetooth LE connections.

Based on the Hardware link-layer implementation, the Bluetooth Low Energy has a higher priority than the Generic FSK protocol and as the effect, the Generic FSK communication is executed during the Idle states (inactive periods) of the Bluetooth LE. The coexistence of the two protocols is handled internally at the Controller level.

The Bluetooth LE part of 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.

The Generic FSK part of the application can either enter in the receive state by double clicking the **SCANSW** button or it can start the periodic transmit by long pressing the **ROLESW** button. It cannot enter in both states at the same time.

The Generic FSK has lower priority than the Bluetooth LE. Therefore, any ongoing Generic FSK receive is paused by the Controller when Bluetooth LE activity is ongoing. The reception is automatically resumed by the Controller when there is no Bluetooth LE activity.

The first Generic FSK transmit command is buffered if there is continuous Bluetooth LE activity (for example, for continuous Bluetooth LE scanning). Any succeeding Generic FSK transmit command indicates failure in the command line interface, if the initial buffered transmit command was not sent yet.

This section describes the implemented profiles and services, user interactions, and testing methods for the Hybrid (Dual-Mode) Bluetooth Low Energy and Generic FSK application.

5.14.1 Implemented profile and services

The Hybrid (Dual-Mode) Bluetooth Low Energy and Generic FSK application implements the GATT client and server for the custom Wireless UART profile and services. It also acts as a Generic FSK transmitter/receptor, repeating a custom packet, at a fixed periodic interval. It uses a predefined identifier, isolated to the address used in the Bluetooth LE protocol of the demo.

- 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 is ready to start either Bluetooth LE scanning for Wireless UART Service, Bluetooth LE advertising Wireless UART Service, Generic FSK periodic transmit of a custom packet or Generic FSK receive, in the available slots not used by the Bluetooth LE protocol.

5.14.2 Supported platforms

The following platforms support the Hybrid (Dual-Mode) Bluetooth Low Energy and Generic FSK application:

- KW45B41Z-EVK
- KW45B41Z-LOC

5.14.3 User interface

After flashing the board, the device is in idle mode (all LEDs flashing). To start Bluetooth LE scanning, press the **SCANSW** button. When in GAP Limited Discovery Procedure of 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.

To start Generic FSK receiving, double click the **SCANSW** button and the device starts receiving packets on the same channel as the Bluetooth LE channel 37, with the network address defined in `gGenFSK_NetworkAddress_c`. The receiver only prints the packets that have the predefined identifier at `gGenFSK_Identifier_c`. To stop Generic FSK reception, double click the **ROLESW** button.

On a different device, start Generic FSK transmit by long pressing the **ROLESW** button. To stop the periodic Generic FSK transmit, long press again the **ROLESW** button, if the transmit procedure is ongoing. The long press **ROLESW** acts as a toggle for the transmit.

See Table 12 for hardware references.

Table 12. Hardware references

Platform	SCANSW	CONNLED	ROLESW
KW45B41Z-EVK	SW2	LED2	SW3
KW45B41Z-LOC	SW2	LED2	SW3

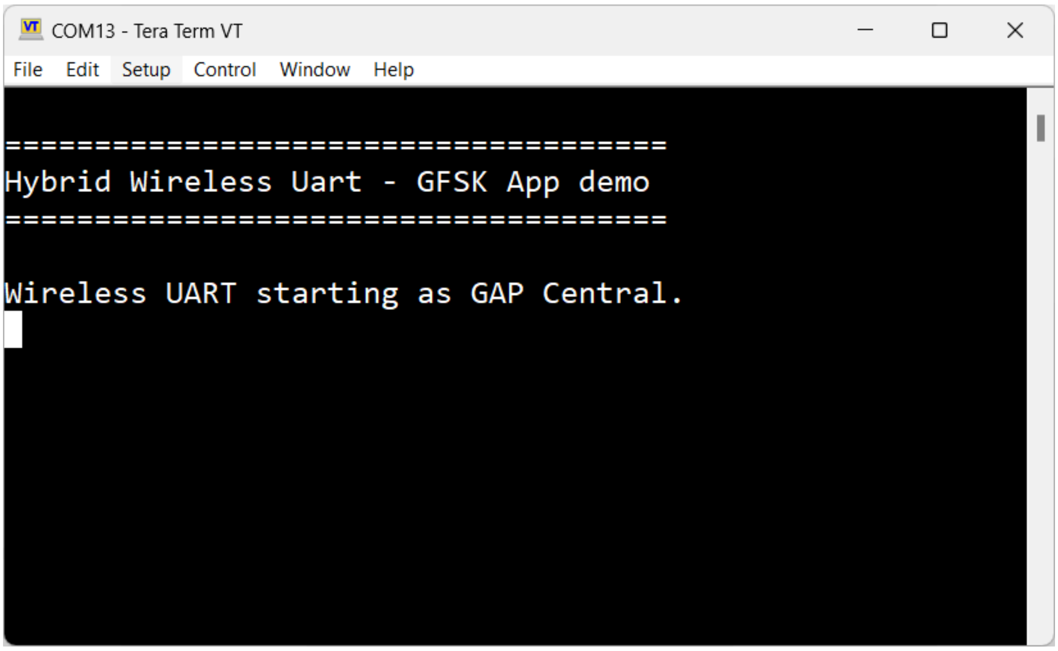
5.14.4 Usage

The application is built to work with another supported platform running the same example. When testing with two boards, perform the following steps:

5.14.4.1 Case 1 (Bluetooth LE)

1. Open a serial port terminal and connect them to the two boards, in the same manner described in in Section 4.3 "Testing devices". The start screen after the board is reset is presented in Figure 52.

Figure 52. Tera Term – Hybrid Wireless UART (Bluetooth LE) - Generic FSK start screen



2. The application starts as a GAP central. To switch the role to a GAP peripheral, press the **ROLESW** button. Depending on the role, when pressing the **SCANSW** button, 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.
4. After creating a connection, the role (central or peripheral) is displayed on the console. The role switch can be pressed again before creating a new connection. An output example can be observed in Figure 53.

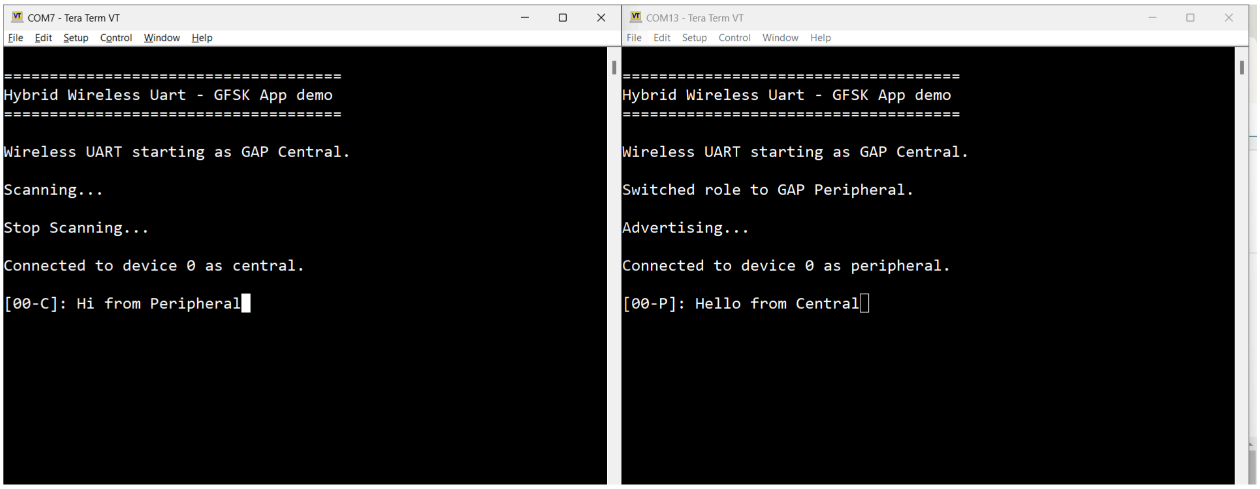


Figure 53. Tera Term – received text on Wireless UART (Bluetooth LE)

5.14.4.2 Case 2 (Generic FSK)

When operating the Generic FSK mode, the following steps should be followed.

1. Open a serial port terminal and connect them to the two boards, in the same manner described in Section 4.3 "Testing devices". The start screen after the board is reset is the same as in Figure 52.
2. The Generic FSK communication direction is not preset. To start receiving, double click the **SCANSW** button on one board. Then, the device starts receiving packets on the same channel as the Bluetooth LE channel 37.
3. To start transmitting, long press the **ROLESW** button on the other board. The transmitting device uses an identifier known by the receiver and its packets are displayed in the CLI as shown in Figure 54:

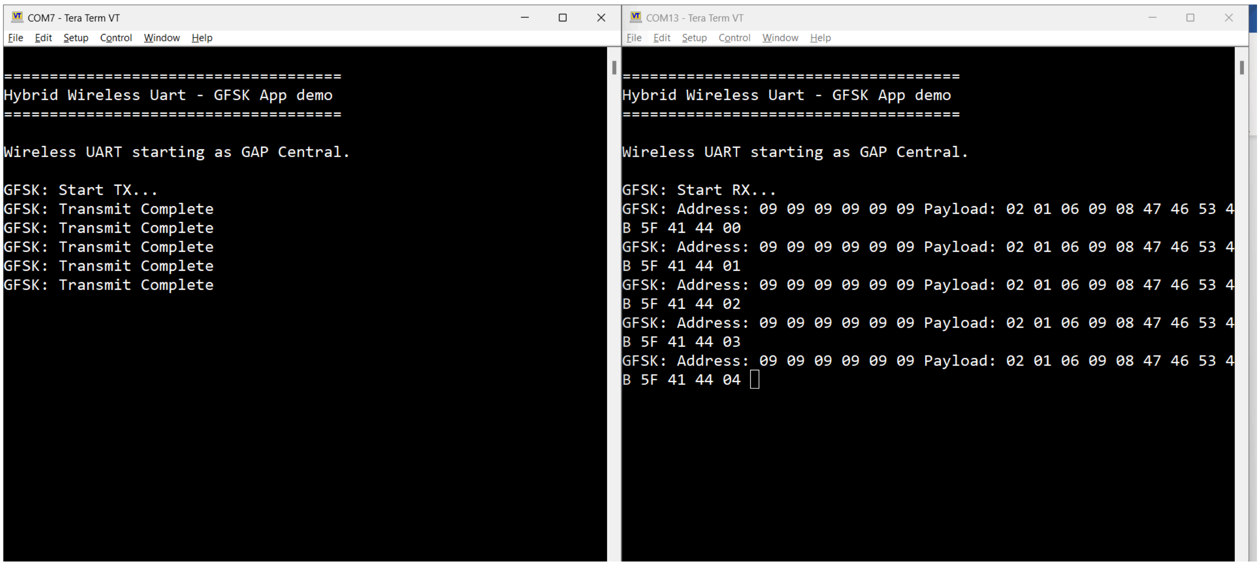


Figure 54. Teraterm – packet transmit/receive on Generic FSK

4. To stop Generic FSK reception, double click the **ROLESW** button.
5. To stop the periodic Generic FSK transmit operation, long press the **ROLESW** button again, if the transmit procedure is ongoing. The long press of the **ROLESW** button acts as a toggle for the transmit. At this point, both devices can reverse the direction of communication by following the exact same steps. See Figure 55.

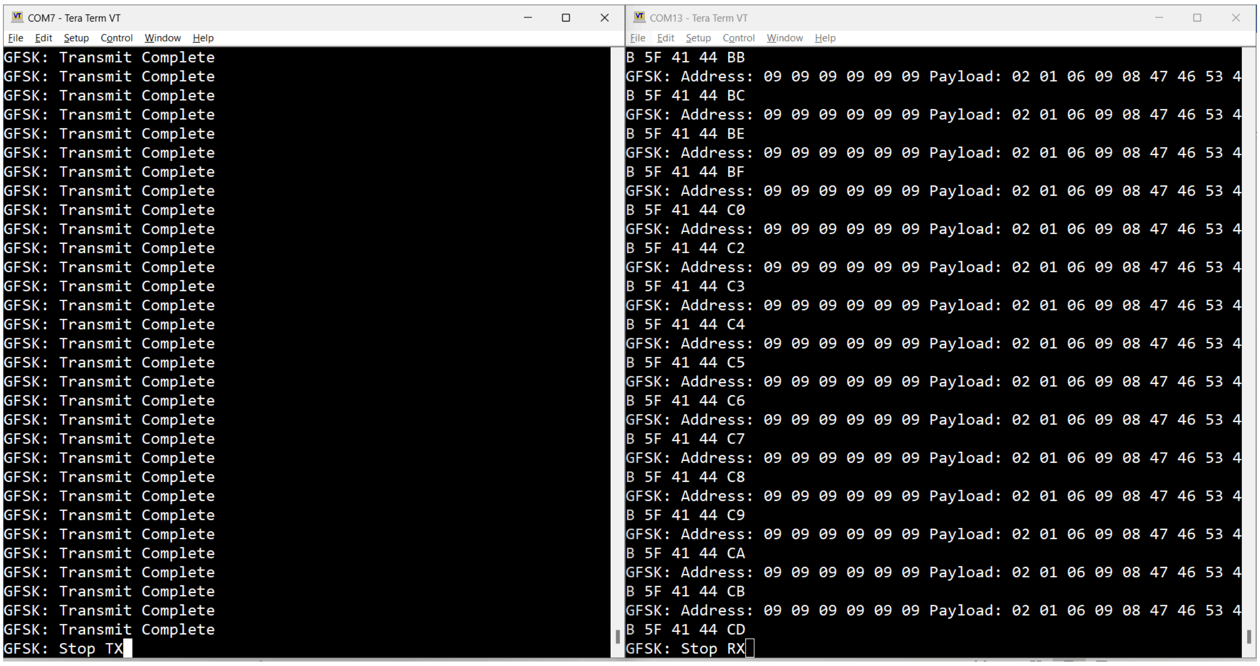


Figure 55. Tera Term – stop Generic FSK activity

5.14.4.3 Case 3 (Dual-mode)

The application is ready to combine both scenarios. It can either:

- Establish a Bluetooth LE connection and perform Generic FSK activity.

- During Generic FSK activity, the Bluetooth LE connection can be established and the Controller pauses receiving or announcing the discarded Generic FSK transmissions. This activity is resumed after the Bluetooth LE activity is finished.
  To run the Dual-mode scenario, follow the steps below:

1. Establish a Bluetooth LE connection as described in Section 5.14.4.1 "Case 1 (Bluetooth LE)".
2. Start Generic FSK activity as described in Section 5.14.4.2 "Case 2 (Generic FSK)", independent of the Bluetooth LE roles chosen in Case 1.
3. Start typing in either of the terminals. As seen in Figure 56, the Bluetooth LE activity is prioritized. In this step, characters are printed in the peer's terminal and Generic FSK continues in the available slots, not used by the Bluetooth LE link.

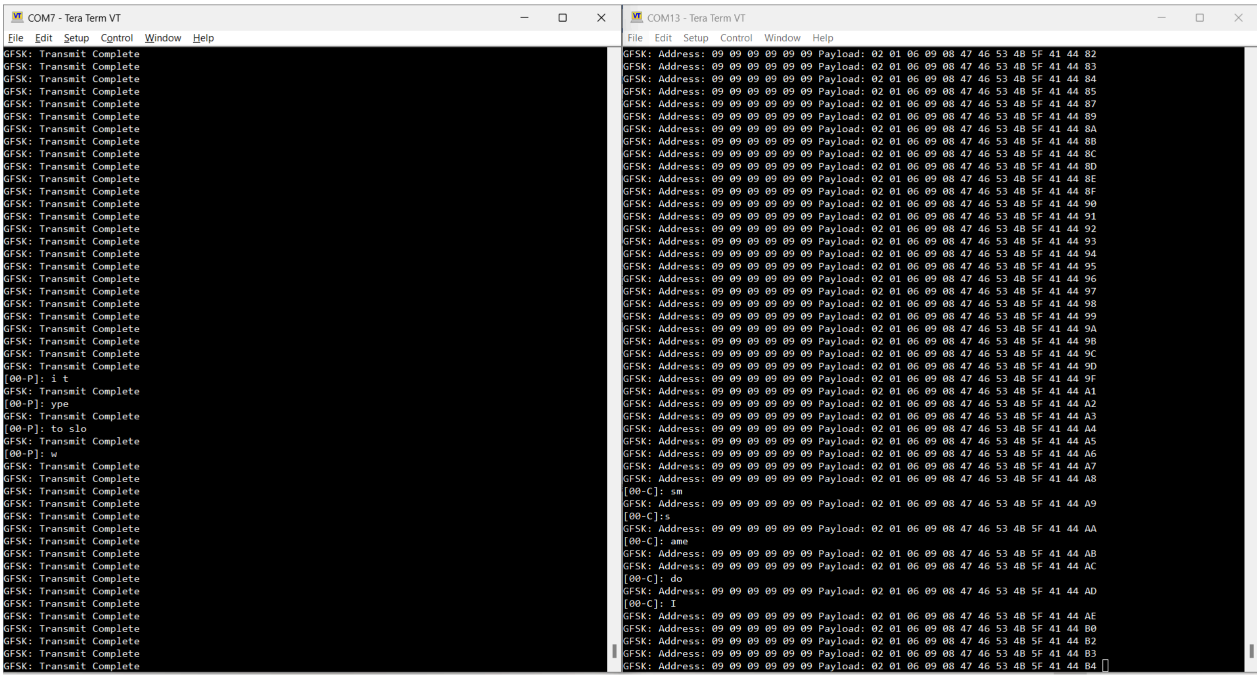


Figure 56. Teraterm – Mixed Wireless UART (Bluetooth LE) and Generic FSK activity

5.14.5 Customization

Use the steps below to change the default settings of this demo.

For Bluetooth LE, the default advertising config (`gAdvParams`) parameter is found in the `app_config.c` file. Also, the scanning parameters (gScanParams) can be found in this file.

Note: *The Generic FSK protocol is active during the inactive periods of the Bluetooth LE protocol. The demo is currently configured to have the scan window equal to the scan interval to make the user aware of this, but this can be changed.*

For Generic FSK, the following defines of interest can be found in genfsk_app.h, described in Table 13 below:

Table 13. Macros (defines)

`gGenFSK_NetworkAddress_c`	This is the network address used for the Generic FSK, in the transmitter payload. It is implicitly set to the 0x8E89BED6, but this can be reconfigured. Ensure that it is also changed on the receiver in the `hybrid_gfsk.c controller` file.
`gGenFSK_H0Value_c`	H0 Value is used in the header.

Table 13. Macros (defines)*...continued*

gGenFSK_Identifier_c	This is the identifier used by the transmitter to be filtered at the receiver. The current implementation filters the Generic FSK packets received, based on this define.
gGenFskApp_Tx Interval_c	This is the interval the transmitter will repeat the transmission of a packet. It is set in milliseconds.

Files of interest

The demo can be found in the `w_uart_genfsk` from the available examples.

The demo is based on the basic Wireless UART with the addition of some Generic FSK files required for working in dual-mode, described in Table 14.

Table 14. File Description

genfsk_app.c	Application common module. Handles the HCI commands and events for the Generic FSK. Sends the events to the application.
genfsk_app.h	Application common module. Exposes public functions.
hybrid_gfsk.c	Controller common module. Handles initialization of Generic FSK.
hybrid_gfsk.h	Controller common module. Exposes public functions.

6 References

For more information, refer to the following documents:

- *Bluetooth Low Energy Application Developer's Guide* (KW45_K32W1_BLEADG)
- *Bluetooth Low Energy Host Stack API Reference Manual* (KW45_K32W1_BLEHSAPIRM)
- *Bluetooth Low Energy Host Stack FSCI* (Framework Serial Connectivity Interface) *API Reference Manual* (KW45_K32W1_BLEHSFSCIAPIRM)
- *Connectivity Framework Reference Manual* (KW45_K32W1_CONNFWRM)
- *Bluetooth Low Energy CCC Digital Key R3 Application Note* (AN12791)

7 Acronyms

The following acronyms are used in this document.

Table 15. Acronyms and abbreviations

Acronym	Description
ANCS	Apple Notification Center Service
ATT	Attribute Protocol
Bluetooth LE	Bluetooth Low Energy
CCC	Car Connectivity Consortium
CCCD	Client Characteristic Configuration Descriptor
DBAF	Decision-Based Advertising Filtering
EATT	Enhanced Attribute protocol
FSCI	Framework Serial Connectivity Interface
GAP	Generic Access Profile
GATT	Generic Attribute Profile
HCI	Host Controller Interface
HID	Human Interface Device
HRS	Heart Rate Server
JSON	Javascript Object Notation
L2CAP	Logical Link Control and Adaptation Protocol
NVM	Non-volatile Memory
OTAP	Over the Air Programming
RF	Radio Frequency
RSSI	Received Signal Strength Indicator
RX	Receive
SDK	Software Development Kit
TX	Transmit
UART	Universal Asynchronous Receiver/Transmitter

8 Note about the source code in the document

Example code shown in this document has the following copyright and BSD-3-Clause license:

Copyright 2022-2024 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

9 Revision history

Table 16 summarizes revisions to this document.

Table 16. Revision history

Document ID	Release date	Description
BLEDAUG v.4.0	26 November 2024	Updated for KW47 EAR 2.1 release Added Section 5.14 "Hybrid (Dual-mode) Bluetooth Low Energy and Generic FSK"
BLEDAUG v.3.6	20 September 2024	Updated for SDK 2.16.100 release Added Section 5.13.7 "Decision-Based Advertising Filtering (DBAF) feature "
BLEDAUG v.3.5	26 June 2024	Updated for SDK 2.16.000 release
BLEDAUG v.3.4	5 April 2024	Updated for SDK 2.15.000 release
BLEDAUG v.3.3	16 October 2023	Updated for MR3 release.
BLEDAUG v.3.2	28 June 2023	Updated for MR2 release
BLEDAUG v.3.1	10 March 2023	Updated for MR1 release
BLEDAUG v.3	8 December 2022	Updated for RFP release, added support for K32W148-EVK board
BLEDAUG v.2	14 September 2022	Updated for PRC 3.0
BLEDAUG v.1	27 June 2022	Updated for PRC 2.0
BLEDAUG v.0	2 March 2022	Initial release for KW45B41Z-EVK platform

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

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 and/or registered trademarks of Arm Limited (or its subsidiaries or affiliates) 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.

Apple — is a registered trademark of Apple Inc.

Bluetooth — the Bluetooth wordmark and logos are registered trademarks owned by Bluetooth SIG, Inc. and any use of such marks by NXP Semiconductors is under license.

IAR — is a trademark of IAR Systems AB.

Kinetis — is a trademark of NXP B.V.

Contents

Date of release: 26 November 2024
Document identifier: BLEDAUG