UG10181

KW47-EVK, MCX-W72-EVK, and FRDM-MCXW72 Bluetooth Low Energy Software Quick Start Guide

Rev. 1.0 — 26 November 2024 User guide

Document information

Information	Content
Keywords	UG10181, Software Quick Start Guide, EVK, (Evaluation Kit), KW47-EVK, MCX-W72-EVK, FRDM-MCXW72, Wireless UART application, NXP IoT Toolbox mobile application, Bluetooth Low Energy (BLE), IAR Embedded Workbench, MCUXpresso IDE, Visual Studio Code
Abstract	This document briefly describes the process of using NXP Bluetooth Low Energy software for the KW47-EVK, MCX-W72-EVK, and FRDM-MCXW72 wireless microcontroller platforms

1 Introduction

This document briefly describes the process of using NXP Bluetooth Low Energy Software for the KW47, MCXW72, and FRDM MCXW72 wireless microcontroller platforms (version 1.1.0). It lists the hardware setup and steps for building and usage of the provided demo applications.

2 Hardware setup

The examples described in this document use a KW47-EVK, MCX-W72-EVK, or a FRDM-MCXW72 as the development platform, as shown in the Figure 1 and Figure 2.

- JLink is the default interface selected in the IAR Embedded Workbench for Arm projects with KW47-EVK, MCX-W72-EVK, or FRDM-MCXW72 platforms included in this release.
- Use jumpers to configure the KW47-EVK or the MCX-W72-EVK board in one of the available power configurations.

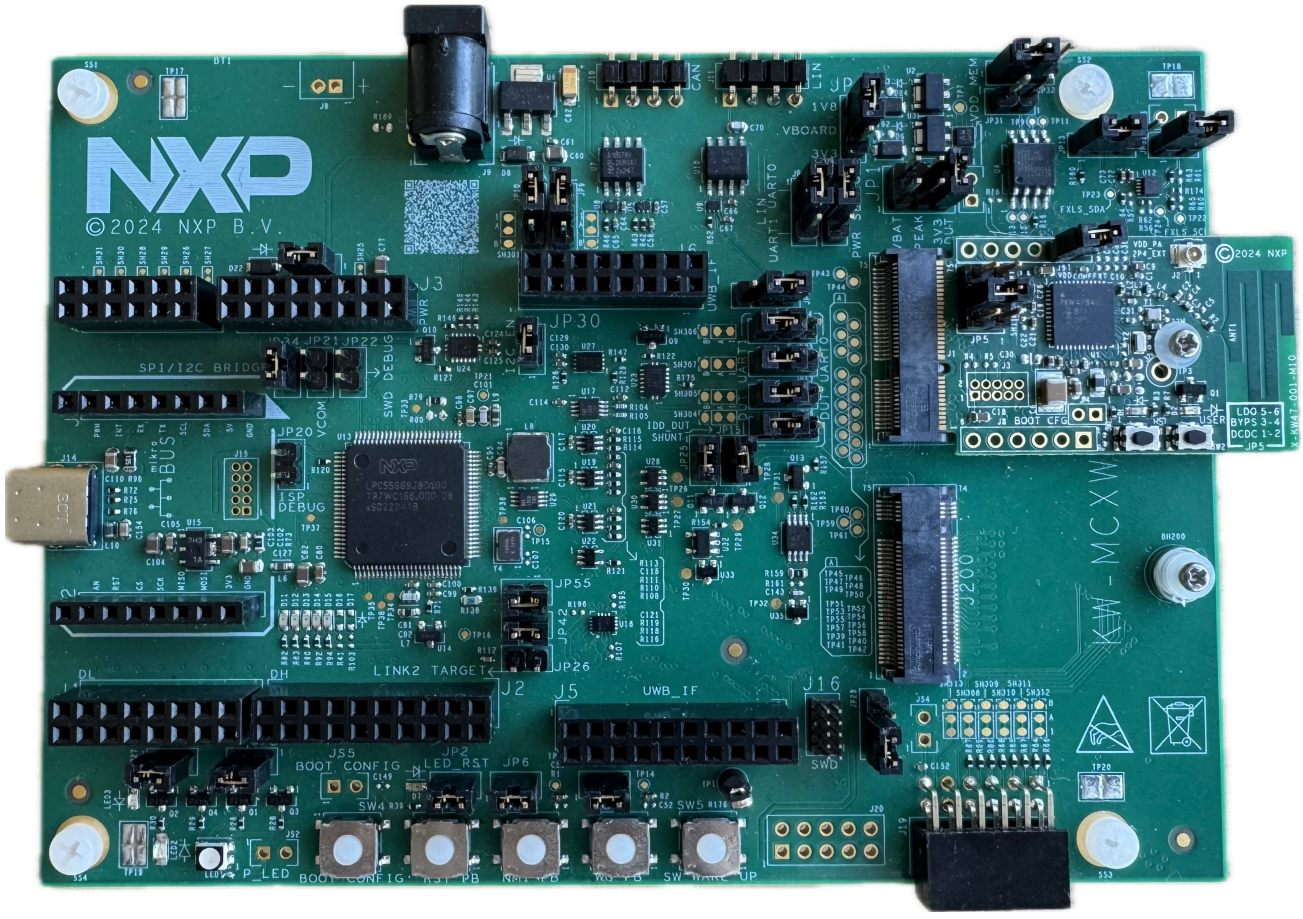

Figure 1. KW47-EVK/MCX-W72-EVK board

Figure 2 shows the FRDM-MCXW72 platform. On the KW47-EVK / MCX-W72-EVK / FRDM-MCXW72 board, the OpenSDA USB port is connected to a Windows PC. The OpenSDA chip on the board requires flashing with appropriate firmware with debugging and virtual serial COM port capabilities.

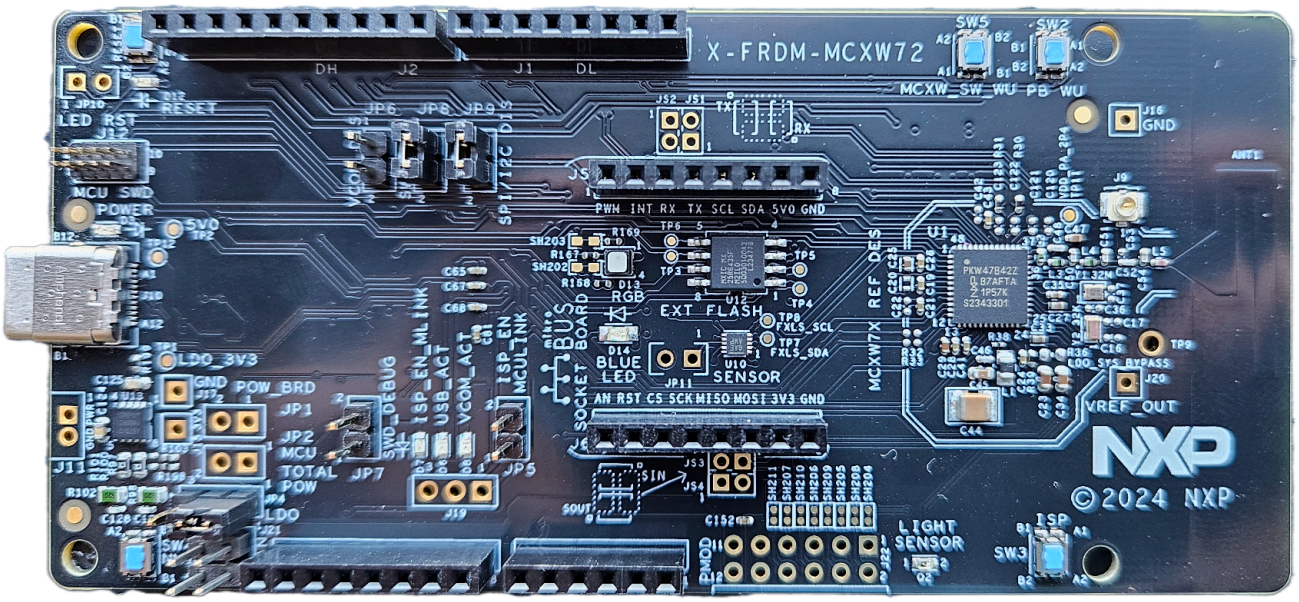

Figure 2. FRDM-MCXW72 platform

Refer to the FRDM-MCXW72 board documentation for the jumper configuration in different power modes.

3 Installing the Connectivity Package

To install the Connectivity Package, configure and download the package archive from the staging system on the https://mcuxpresso.nxp.com website. You can simply download the precreated package archive if it is available on the same website.

Note: *Use the default location for the package (`C:\NXP`) and create a subfolder there specific to each device and release.*

Note: *Prior to loading any wireless SDK example, update your NBU image with the provided binaries in the following folder of the SDK: `../middleware/wireless/ble_controller/bin`*

4 Building the binaries

This section describes the necessary steps for obtaining the binary files for usage with the boards.

4.1 Prerequisites

To build any of the demo applications, you need the following toolchain:

- IAR Embedded Workbench for Arm (details in release note)
- MCUXpresso IDE (details in release note)
- Visual Studio Code with "*MCUXpresso for Visual Studio Code*" extension (details in release note)
- Teraterm (version 4.105 or higher)

The Connectivity Software Package does not include support for any other toolchains. The packages must be built with the debug configuration to enable debugging information. This package includes various sample applications that can be used as a starting point.

4.2 Conventions for building the *wireless_UART* application.

The following sections present the steps required for building the *wireless_UART* application. All applications can be found using the following placeholders for text:

- `<connectivity_path>`: represents the root path for the SDK.
- `<board>`: represents the target board for the demo app, “`kw47evk`” in this case.
- `<RTOS>`: represents the scheduler or RTOS used by the app; it can be either “`bm`” or “`freertos`”.
- `<demo_app>`: represents the demo application name.
- `<IDE>`: represents the integrated development environment used to build projects; “iar” in this case.
- `<core_id>`: represents the target CPU on which the application will run, "`cm33_core0`" in this case.
  The demo applications general folder structure is the following:
  `<connectivity_path>\boars\<board>\wireless_examples\bluetooth\<demo_app>\<core_id>\<RTOS>\<IDE>`

Selected application: w_uart

Board: kw47evk or frdmmcxw72 or mcxw72evk

RTOS: FreeRTOS

Resulting location:

`<connectivity_path>\boards\<kw47evk or mcx-w72-evk or frdmmcxw72>\wireless_examples\bluetooth\w_uart\freertos\<IDE>`

4.3 Building and flashing the BLE software demo applications using IAR Embedded Workbench

Use the following steps in order to build and flash the BLE software demo applications using the IAR Embedded Workbench:

1. First unpack the contents of the archive to a folder on the local disk. Then, navigate to the resulting location starting from the SDK root directory.
2. Open the IAR workspace file (`*.eww` file format) highlighted file in Figure 3.

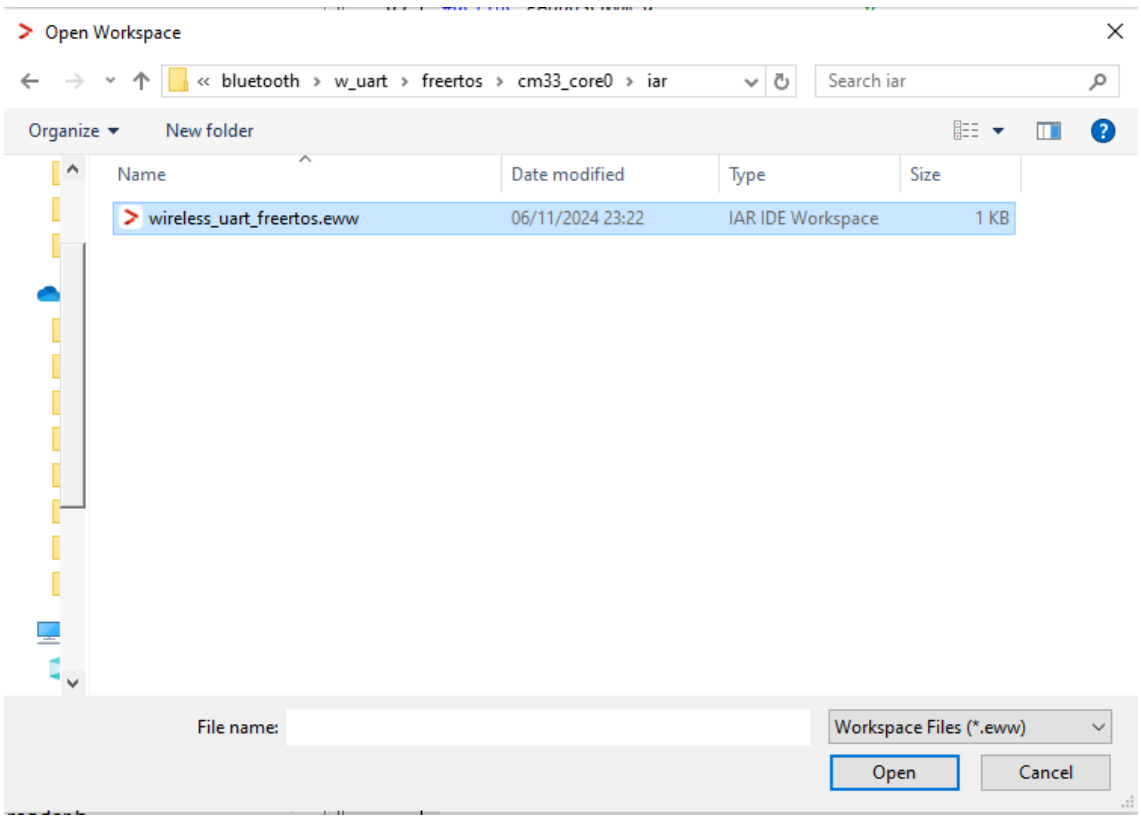

Figure 3. Wireless UART IAR demo project location

3. Choose between Debug and Release configurations in the drop-down selector above the project tree in the workspace, as seen in Figure 4.

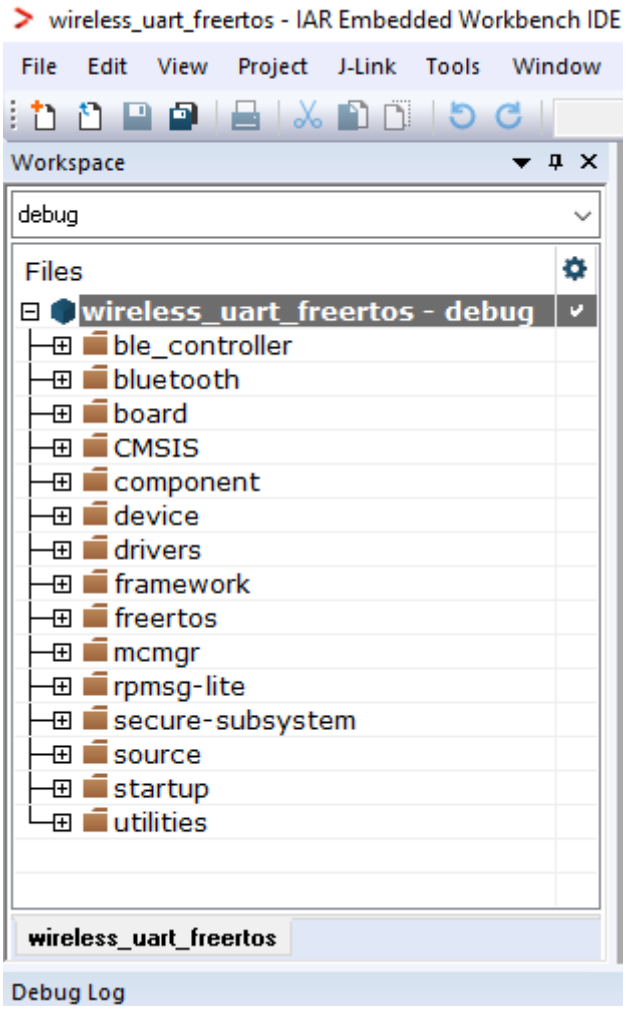

Figure 4. Select the desired configuration (Debug or Release)

Figure 5 shows the Wireless UART - IAR workspace.

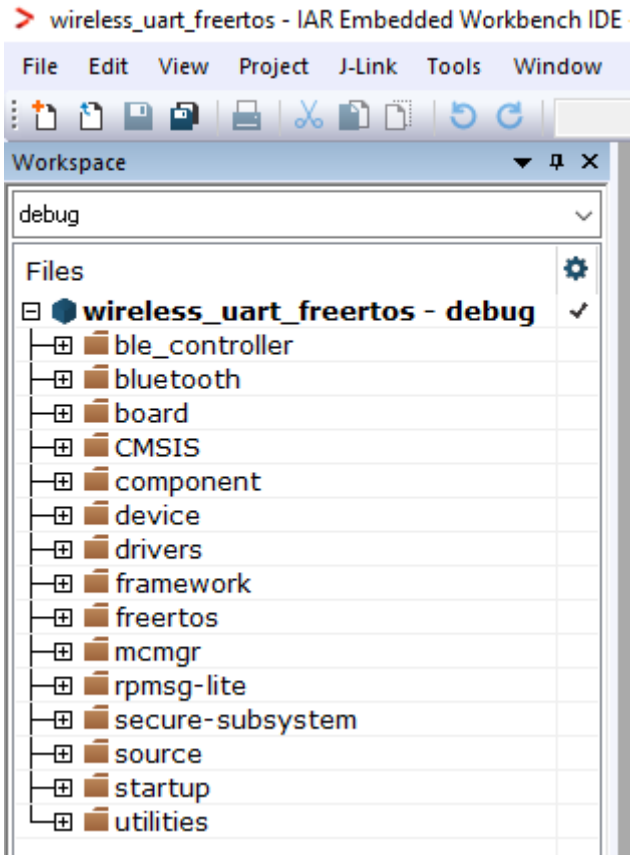

Figure 5. Wireless UART - IAR workspace

4. Build the Wireless UART project using the options shown in Figure 6.

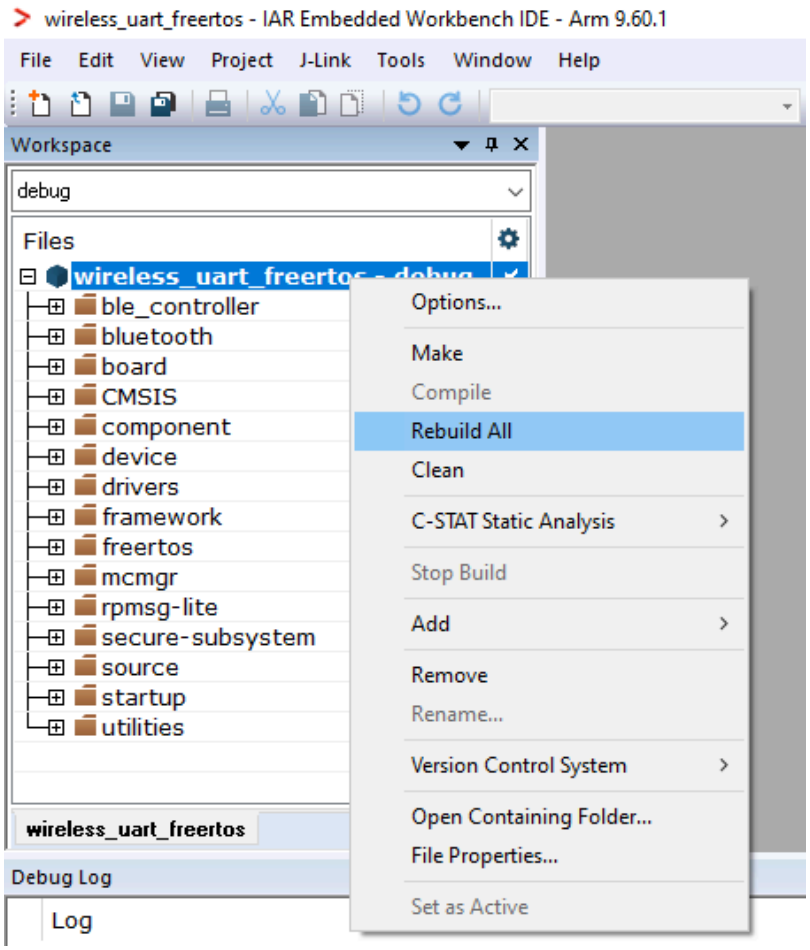

Figure 6. Build Wireless UART application

5. Make the appropriate debugger settings in the project options window, as seen in Figure 7.
   Project > Options (Alt+F7) > Debugger > Setup (tab) > Driver > J-Link/J-Trace

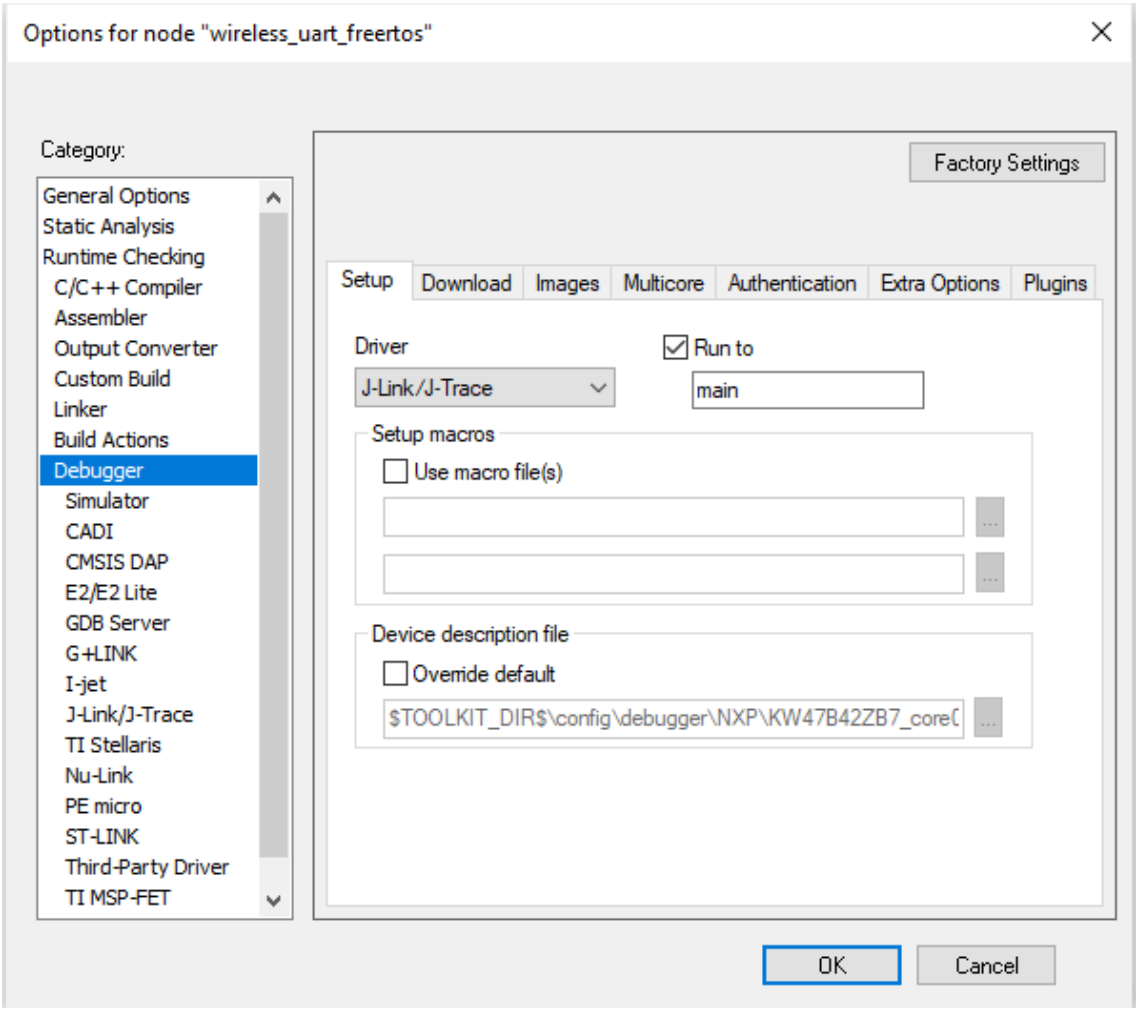

Figure 7. Debugger Settings for the Wireless UART project

6. Click the "**Download and Debug**" button (or **CTRL+D**) to flash the executable onto the board, as seen in Figure 8.

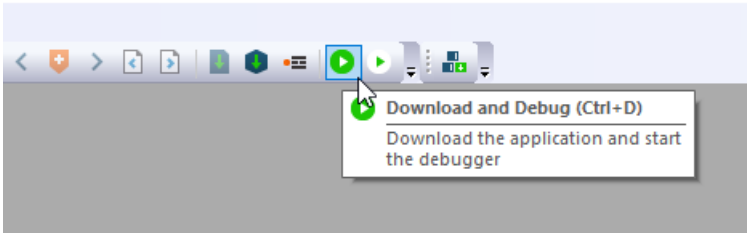

Figure 8. Download and Debug the Wireless UART application

7. Press **Go** (**F5**). At this moment, the board starts running the application as shown in Figure 9.

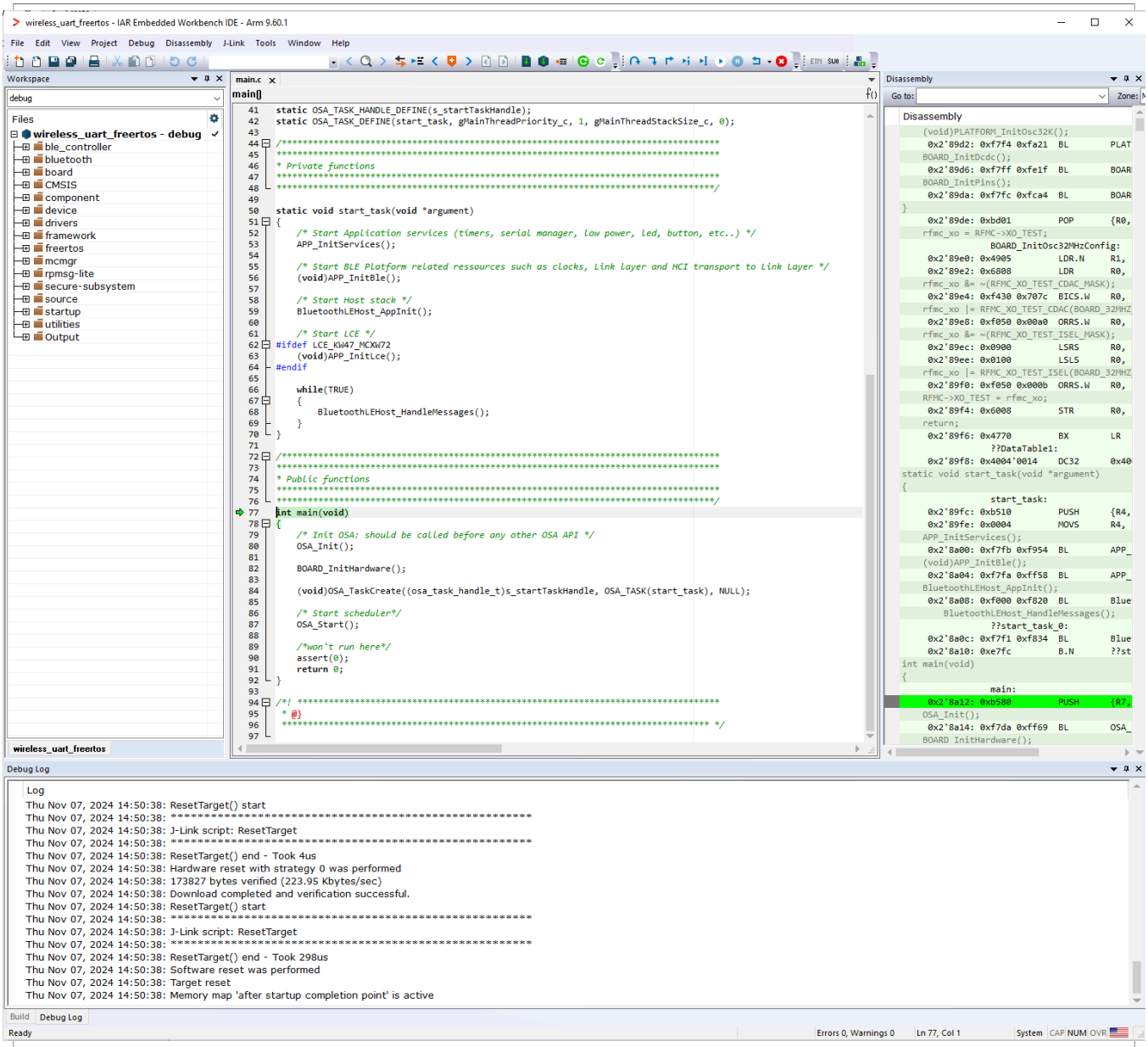

Figure 9. Running the code on IAR

4.4 Building and flashing the BLE Software Demo applications using MCUXpresso IDE

To build and flash the BLE software demo applications using MCUXpresso IDE, follow the steps listed below:

1. Open MCUXpresso IDE and open an existing or new workspace location.

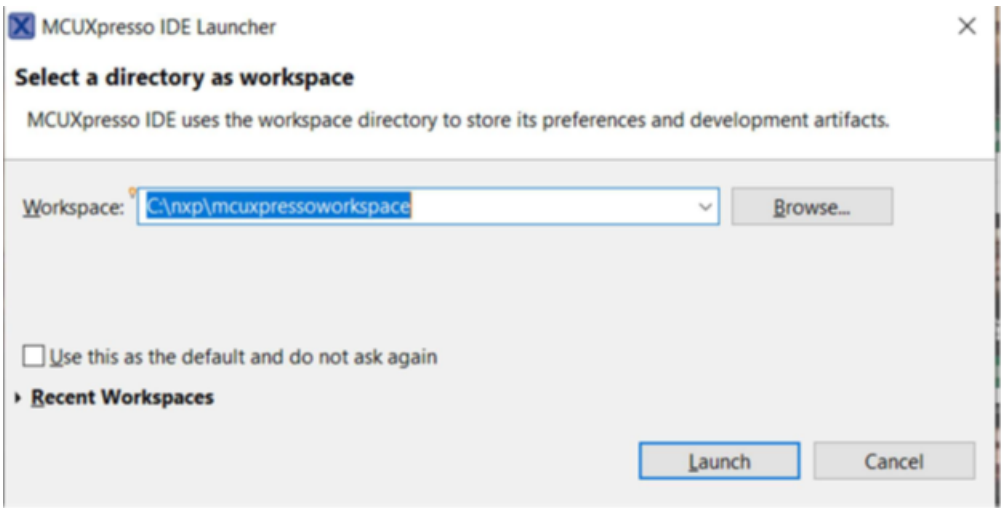

Figure 10. Select MCUXpresso IDE workspace

2. Drag and drop the package archive into the **MCUXpresso Installed SDKs** area in the lower right of the main window.

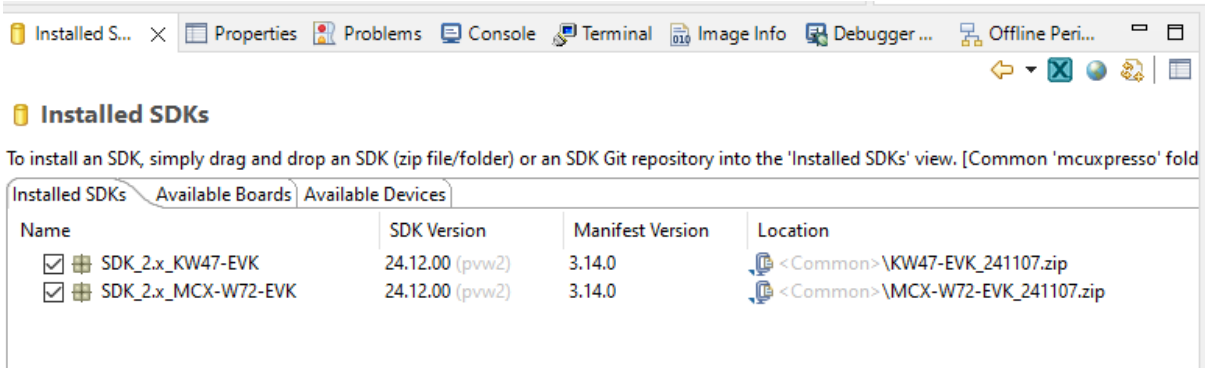

Figure 11. Installed SDKs in MCUXpresso IDE workspace

3. After the SDK is loaded successfully, select the "**Import the SDK examples(s)…**" to add examples to your workspace.

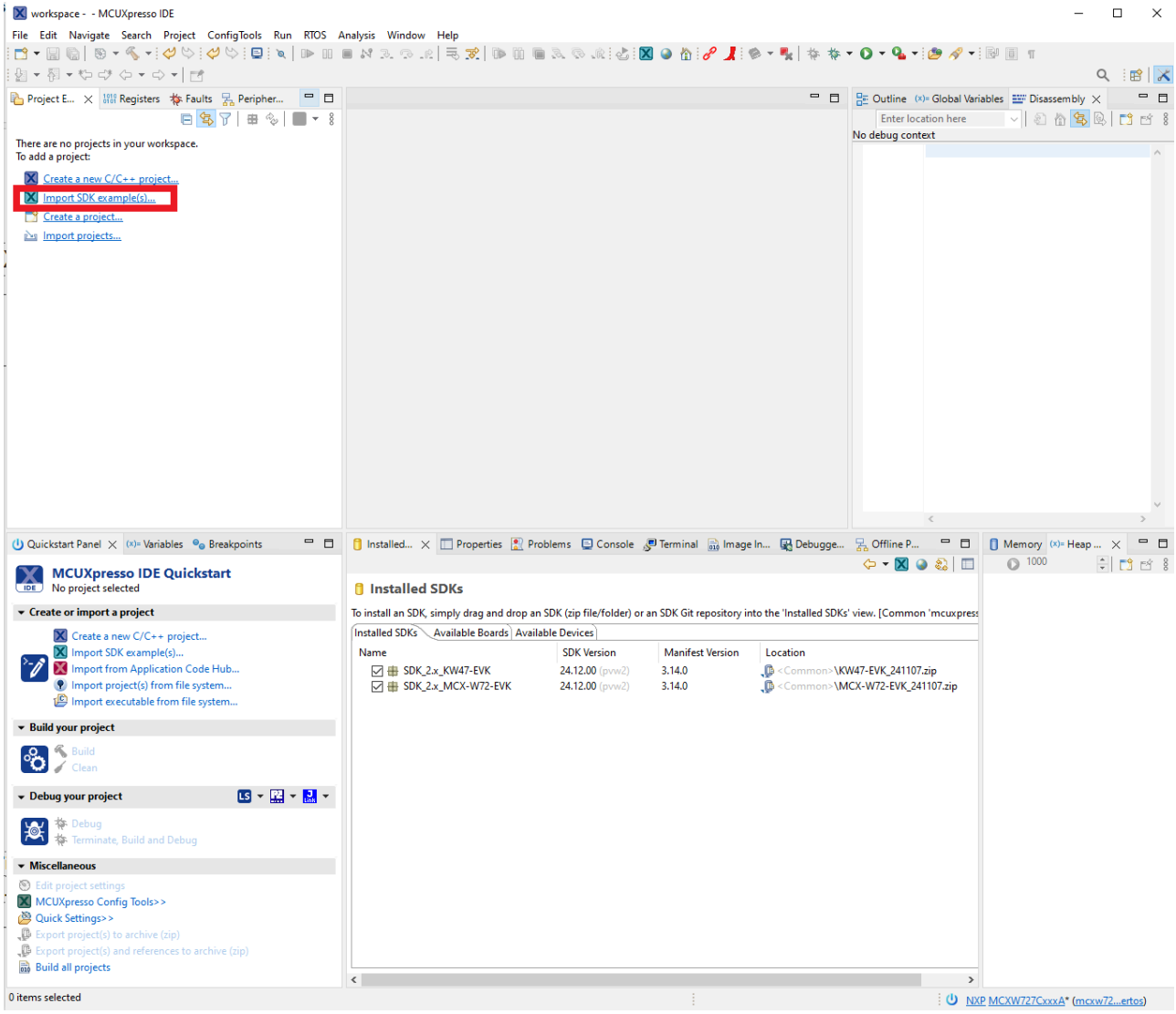

Figure 12. Importing SDK example(s)

4. To select the desired example(s), select the KW47-EVK, MCX-W72-EVK, or FRDM-MCXW72 board and then click the “**Next**” button:

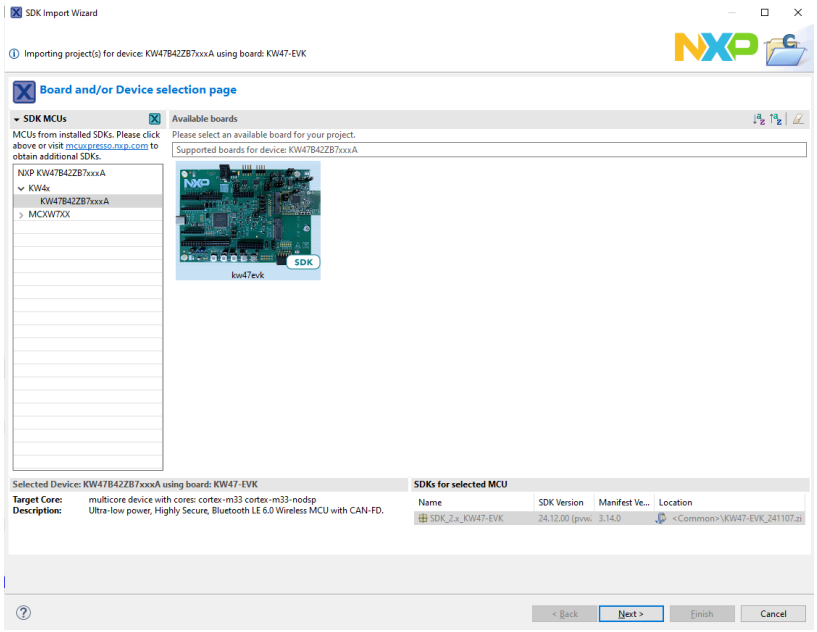

Figure 13. Selecting the KW47-EVK board

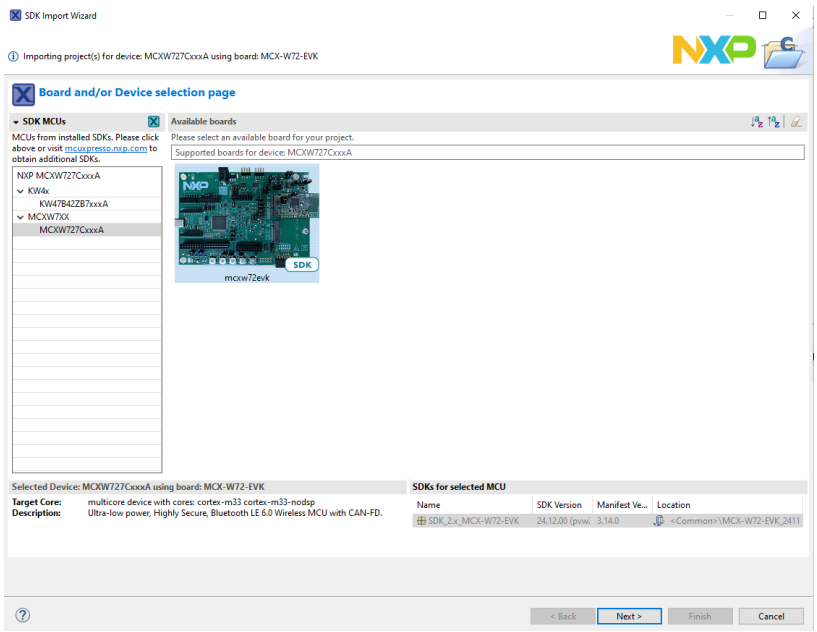

Figure 14. Selecting the MCX-W72-EVK board

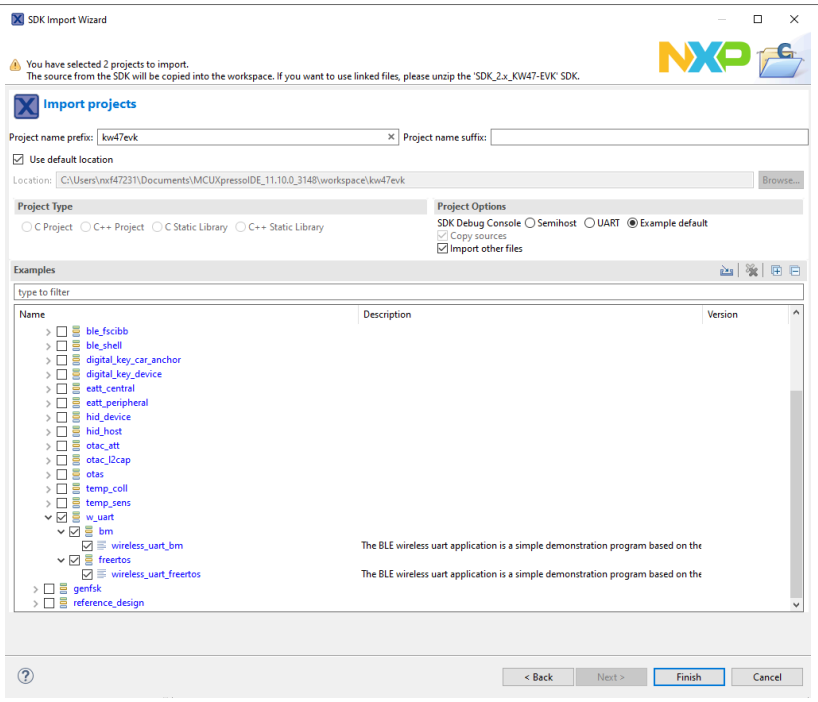

Figure 15. Select Wireless UART FreeRTOS project

5. Build the `wireless_uart_freertos` project.

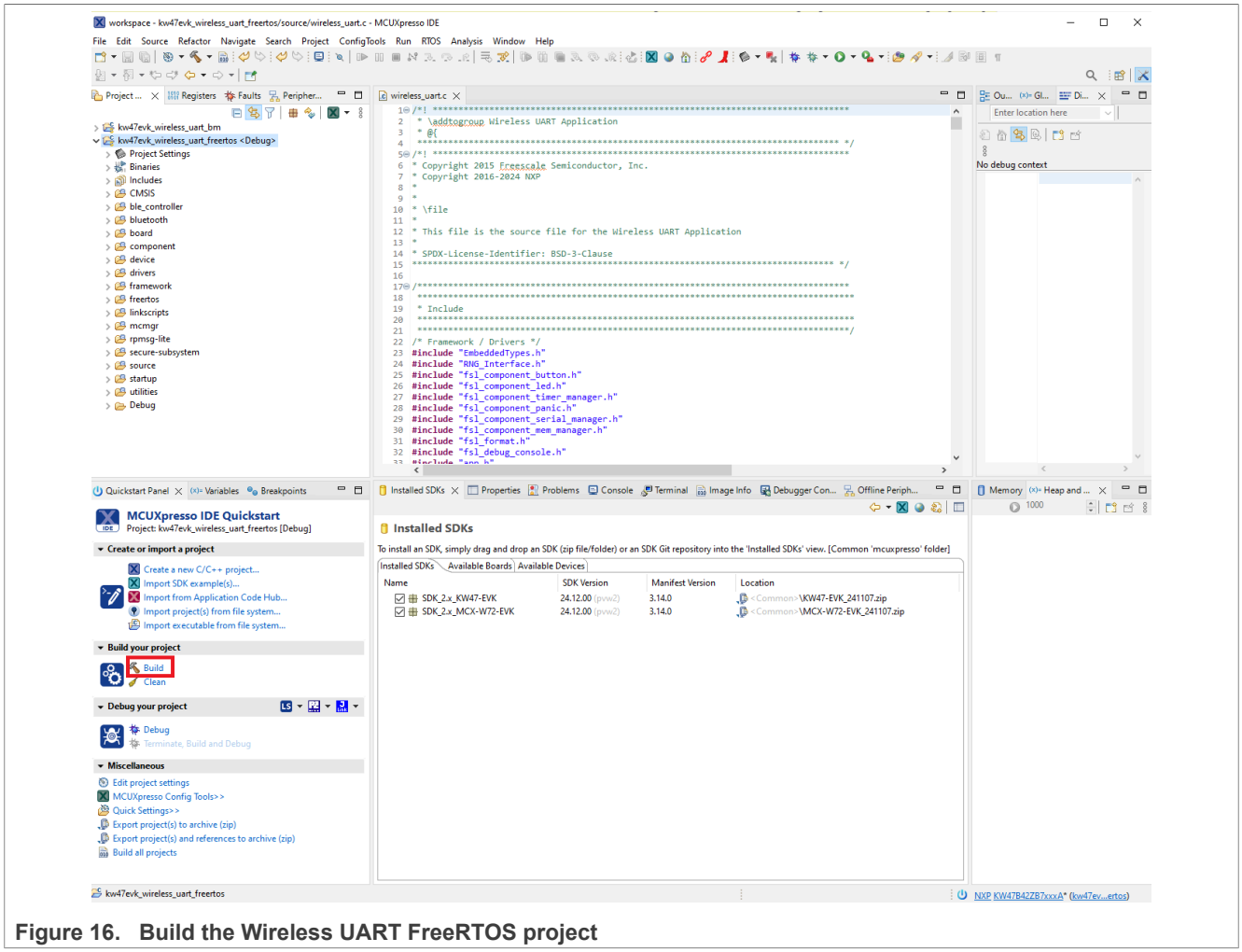

Figure 16. Build the Wireless UART FreeRTOS project

6. Click the "**Debug**" button to download the executable onto the board. Make sure you select the appropriate device to flash.

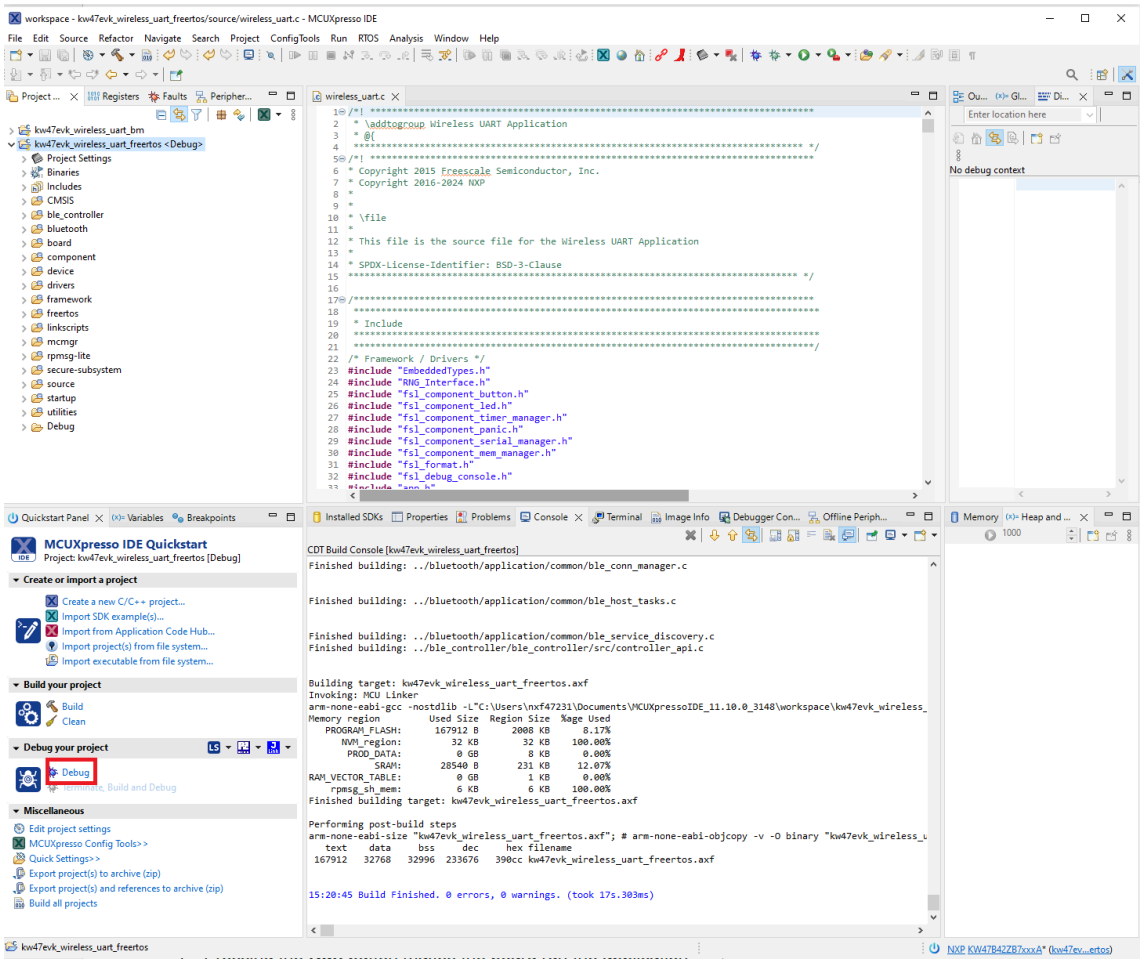

Figure 17. Download and debug the Wireless UART FreeRTOS project

7. Pressing the **Run** button makes the board run the application.

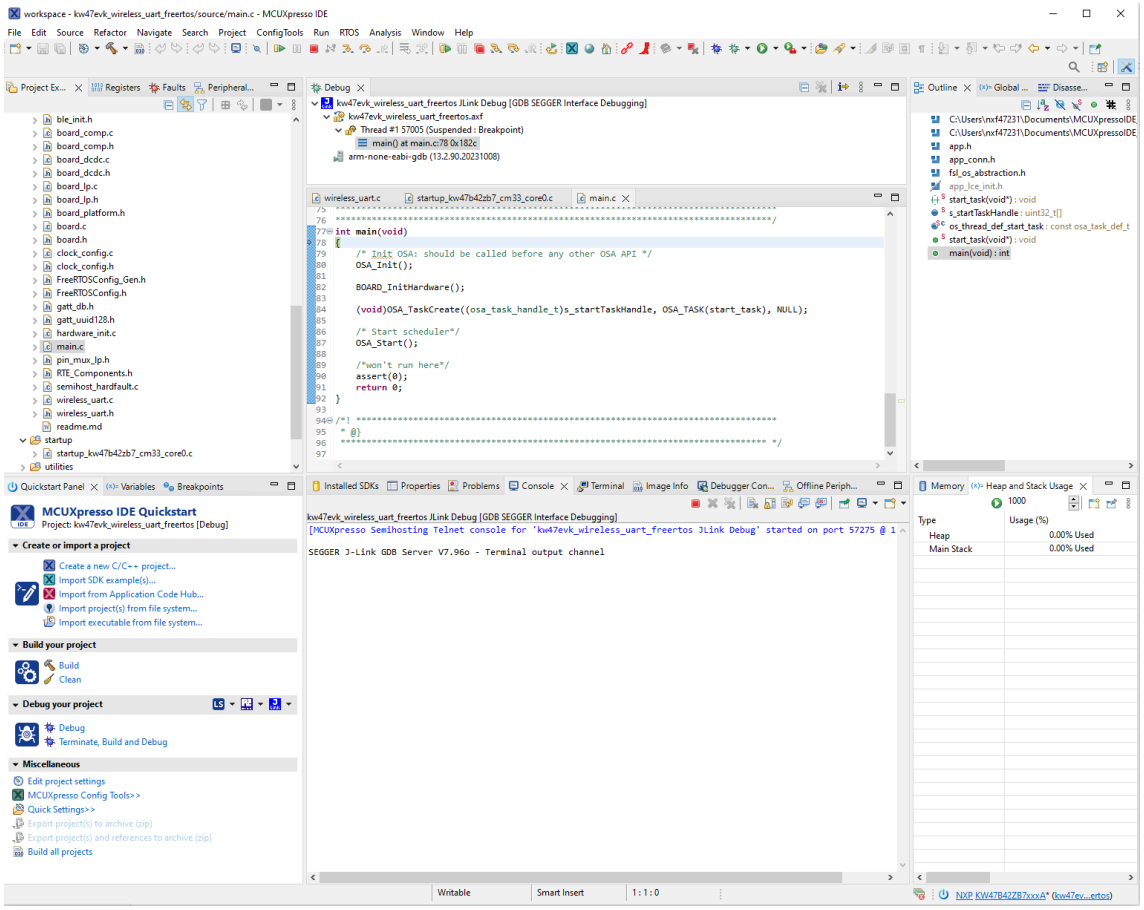

Figure 18. Running the code on MCUXpresso IDE

4.5 Building and flashing the BLE software demo applications using Visual Studio Code

To build and flash the BLE software demo applications using Visual Studio Code, follow the steps listed below:

1. Open Visual Studio Code and open the MCUXpresso for Visual Studio Code extension.

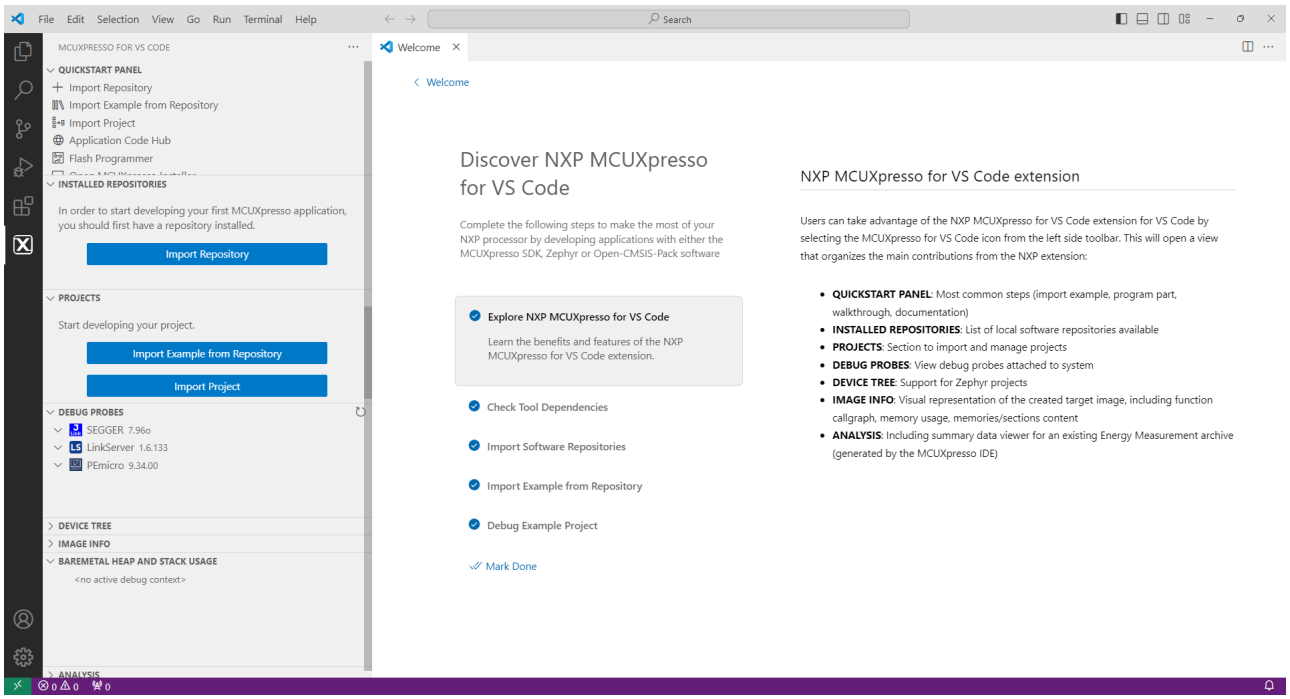

Figure 19. MCUXpresso for Visual Studio Code extension

2. Import the SDK package: click “**Import Repository**”. Then, choose the “**Local**” option (if the SDK is archived use “**Local archive**”), browse to the path of the SDK you want, and click “**Import**”.

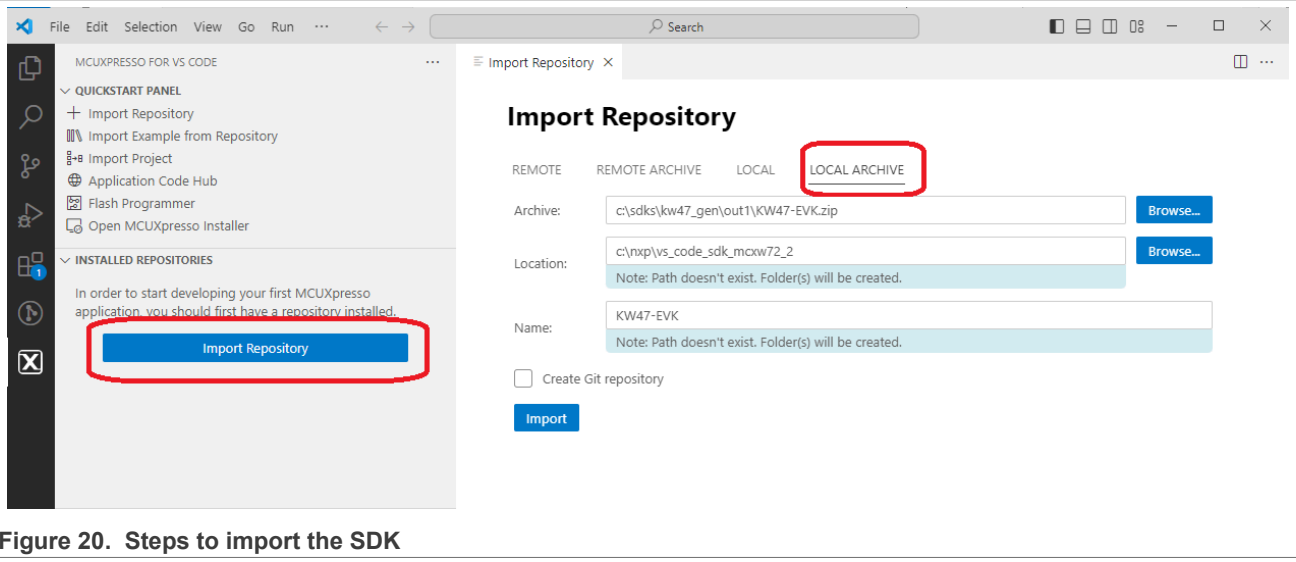

Figure 20. Steps to import the SDK

3. After the SDK is loaded successfully, select the “**Import Example from Repository**” to add an application to your workspace. Choose the repository, toolchain (Arm GNU), board, example you want to add, and the location where the VS Code project would be created. Then click “**Create**”.

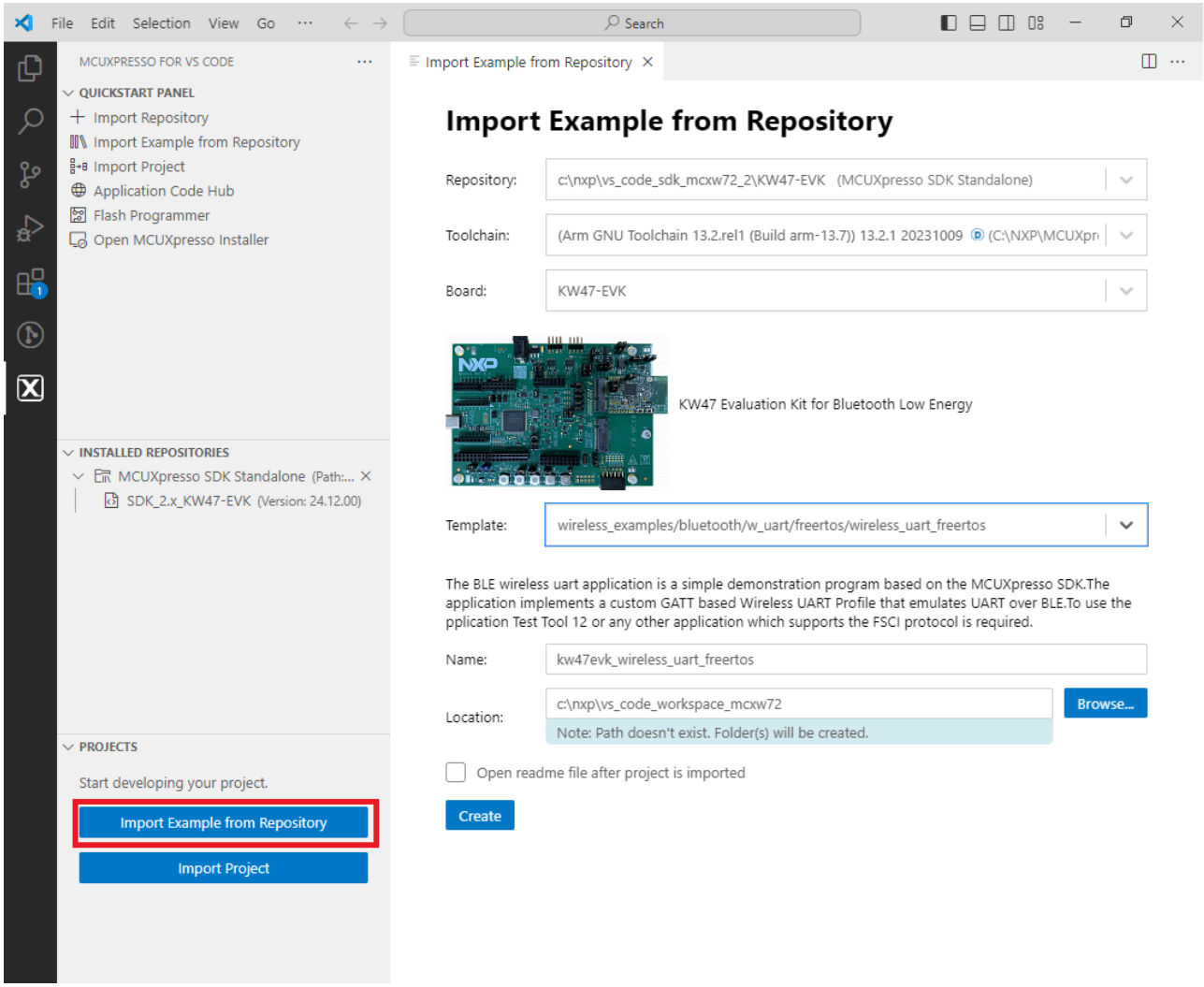

Figure 21. Steps to import the example application

4. The application now appears in the “**Projects**” tab on the left. Build the application by pressing the “**Build selected**” button.

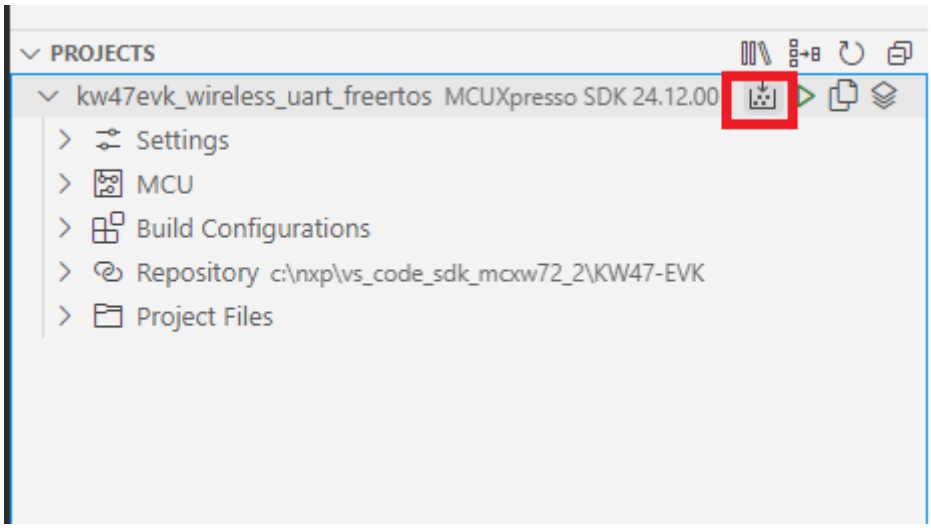

Figure 22. Building the Wireless UART FreeRTOS application

5. After the build is completed successfully, debug the application by clicking the "**Debug**" button.

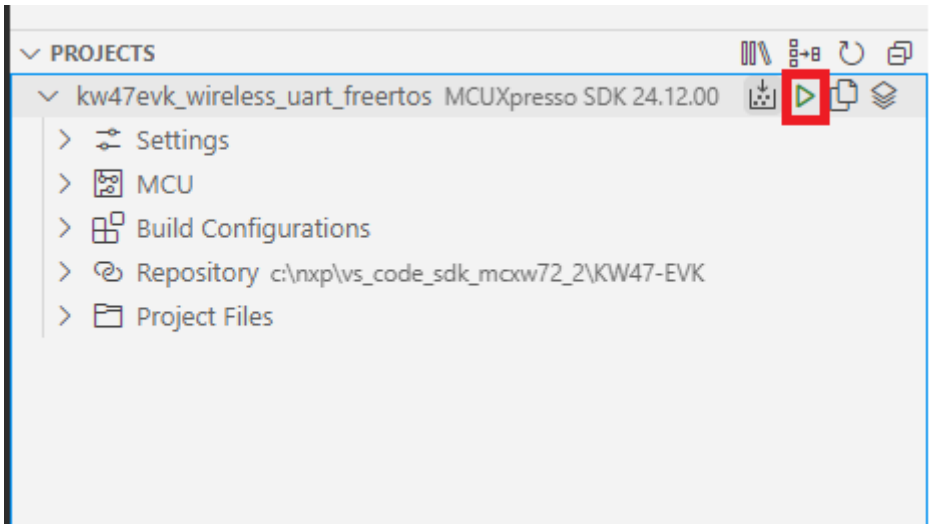

Figure 23. Debug the Wireless UART FreeRTOS application

6. Press the run ("**Continue**") button twice to run the application on the board.

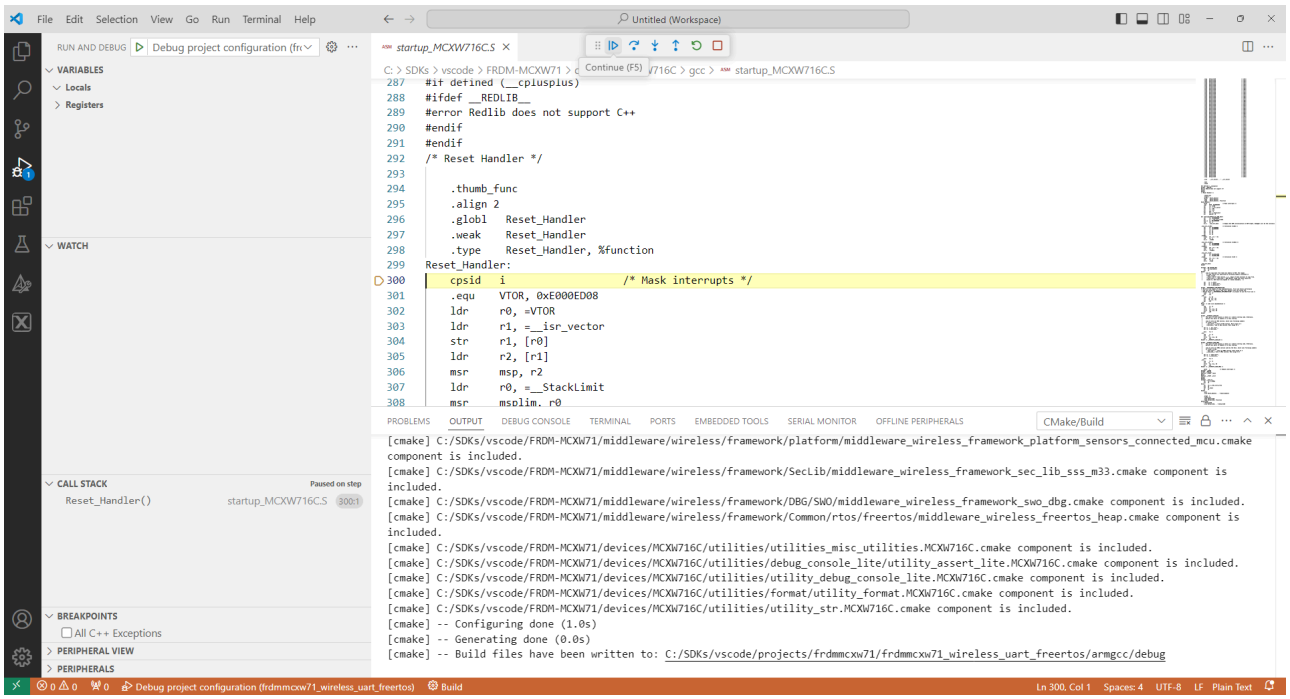

Figure 24. Running the Wireless UART FreeRTOS application

5 Running the Wireless UART application using NXP IoT Toolbox mobile application

To run the Wireless UART application using NXP IoT Toolbox mobile application, follow the steps as listed below:

Note: *Before working on these steps, ensure to install the latest version of the NXP IoT Toolbox mobile application from the application store.*

1. Flash the board with the Wireless UART application as previously described.
2. Open Tera Term (or any other Serial Communication software) and choose **Serial**. Then choose the port of your board and click **OK**. See Figure 25.

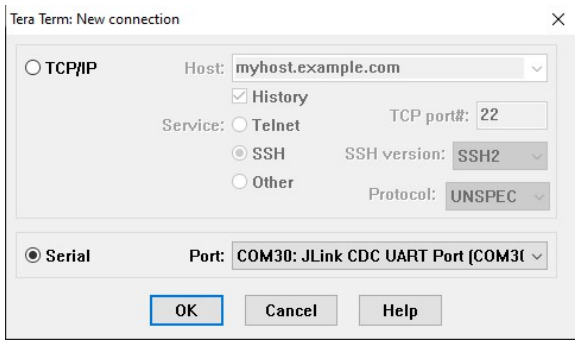

Figure 25. Setting up a new Tera Term connection for Wireless UART application

3. Stop running the code on the board. Go to **Setup >Serial Port**. See Figure 26.

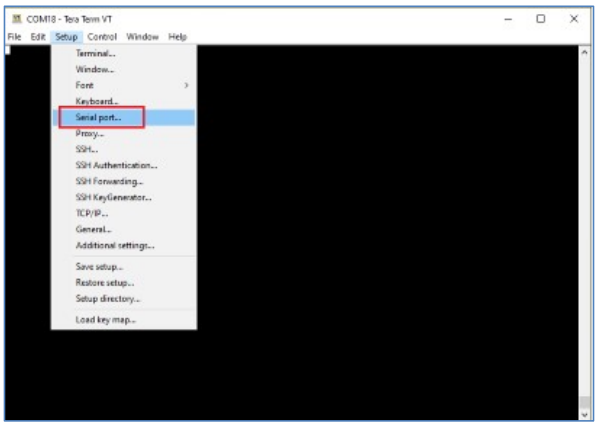

Figure 26. Setting the serial port

4. Choose the required Speed (baud rate) and Flow Control values. Here, the baud rate should be 115200 and the Flow control mode should be set to **Xon/Xoff**.

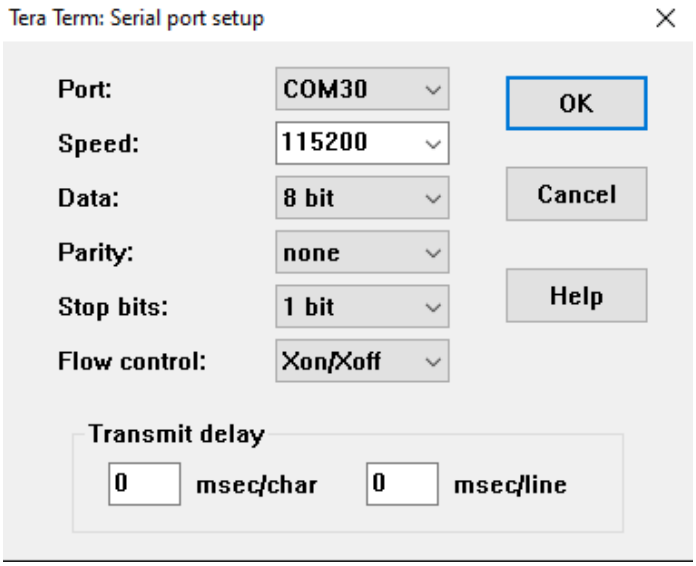

Figure 27. Setting the baud rate

5. Ensure that the Bluetooth device of your phone is enabled.
6. Open the *IoT Toolbox* application and select the ***Wireless UART*** icon, as shown in Figure 28.

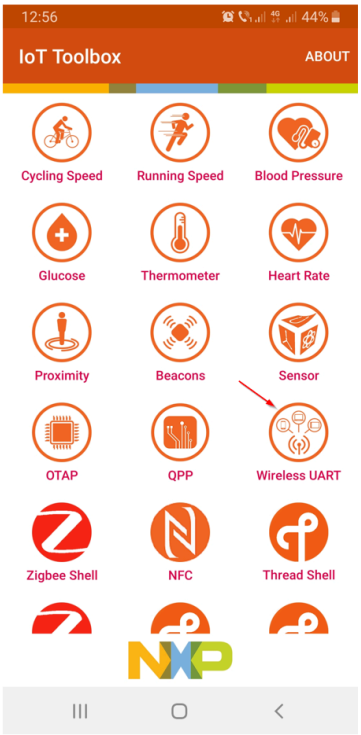

Figure 28. NXP IoT Toolbox application

7. Now run the application on the board. On this step, the RGB LED blinks white quickly. Press the button **SW3** to switch to Peripheral (Central) mode, then press **SW2** to start Advertising (scanning). The Tera Term terminal shows the message shown in Figure 29.

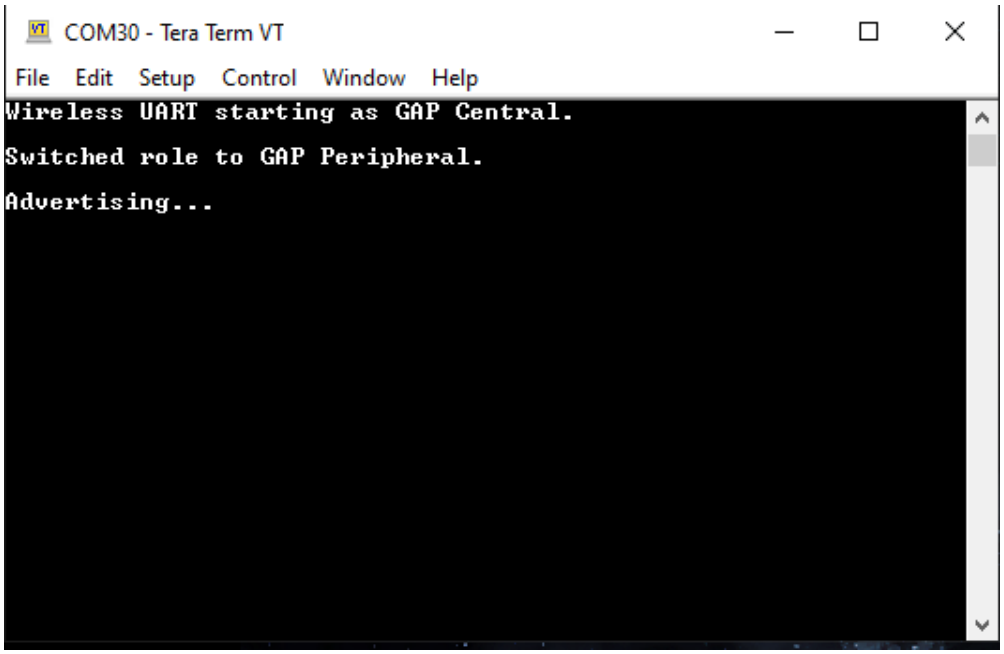

Figure 29. Tera Term console

8. The device should become visible for the Wireless UART mobile application, as shown in Figure 30.

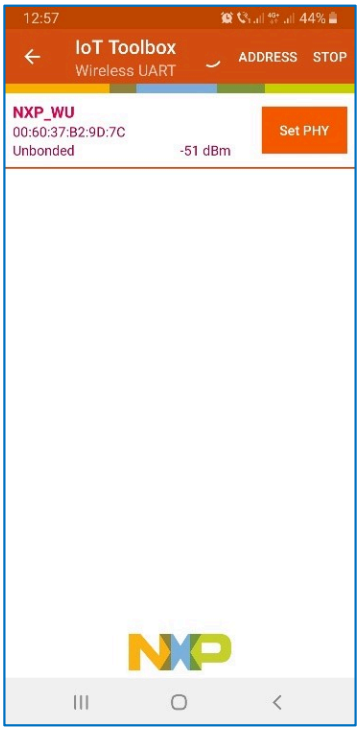

Figure 30. NXP Wireless UART scanned by IoT toolbox app

9. Select the device that appears in the **Wireless UART** tab to connect to it. After connecting, the mobile application shows the console, as shown below. Users can choose between the Wireless Console and Wireless UART tabs.
   The device should become visible for the Wireless UART mobile application, as shown in Figure 31.

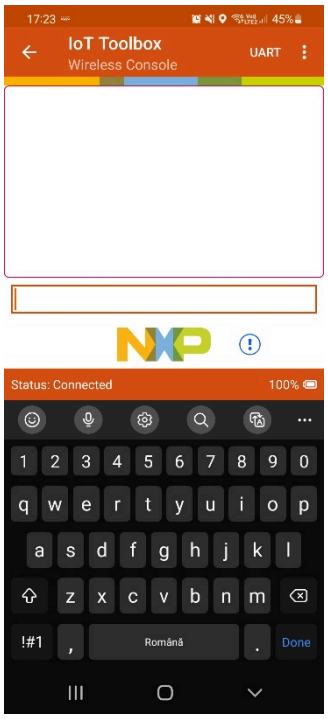

Figure 31. Wireless UART working in Connected mode

10. On the IoT Toolbox Wireless application console, the message is introduced in one line and sent to the peer. If Wireless UART mode is used, communication is done character by character. On Tera Term, the message received is displayed. One character at a time can be sent from Tera Term to the peer. When the mobile application receives the character, it displays it.
    The Figure 32 shows a KW47-EVK / MCX-W72-EVK / FRDM-MCXW72 board acting as a peripheral, connected to a phone using IoT Toolbox in Wireless Console and Wireless UART modes.

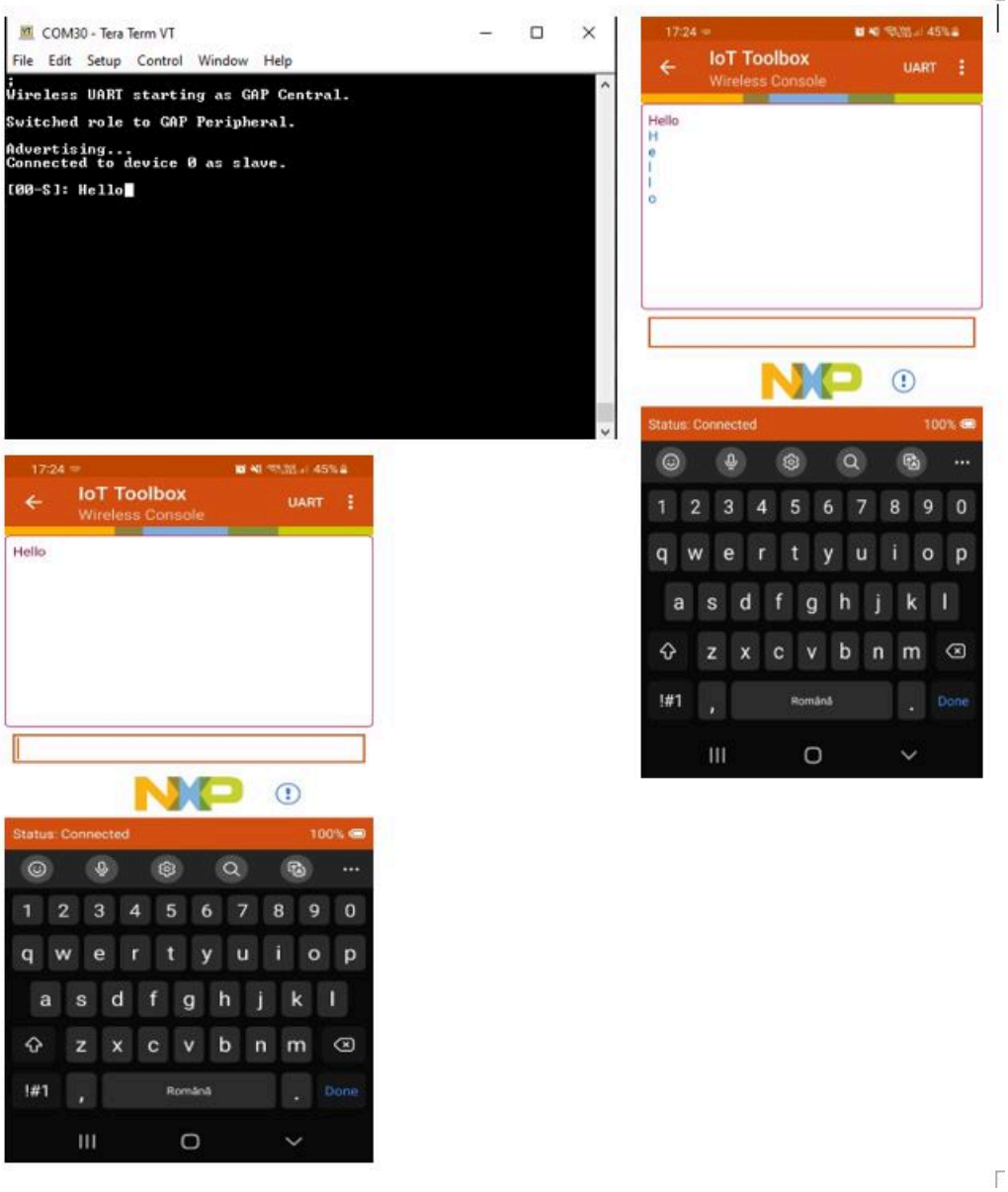

Figure 32. Exchanging messages between the phone and Wireless UART Console

6 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 will have lower priority than the Bluetooth LE, therefore any ongoing Generic FSK receive will be paused by the Controller when Bluetooth LE activity is ongoing and automatically resumed by the Controller when there is no Bluetooth LE activity.

The first Generic FSK transmit command will be buffered if there is continuous Bluetooth LE activity (example is continuous Bluetooth LE scanning). Any succeeding Generic FSK transmit command will indicate 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.

6.1 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 and will also act as a Generic FSK transmitter/receptor, repeating a custom packet, at a fixed periodic interval, with 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.

6.2 Supported platforms

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

- KW45B41Z-EVK
- KW45B41Z-LOC

6.3 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 will start receiving packets on the same channel as the Bluetooth LE channel 37, with the network address defined in gGenFSK_NetworkAddress_c. The receiver will only print the packets that have the predefined identifier at gGenFSK_Identifier_c. To stop Generic FSK receiving 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 will act as a toggle for the transmit.

See Table 1 for hardware references.

Table 1. Hardware references

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

6.4 Usage

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

Case 1 (Bluetooth LE):

1. Open a serial port terminal and connect them to the two boards. (For the procedure, refer to the Section "Testing devices" in *Bluetooth Low Energy Demo Applications User Guide*. The start screen after the board is reset is presented in Figure 34.

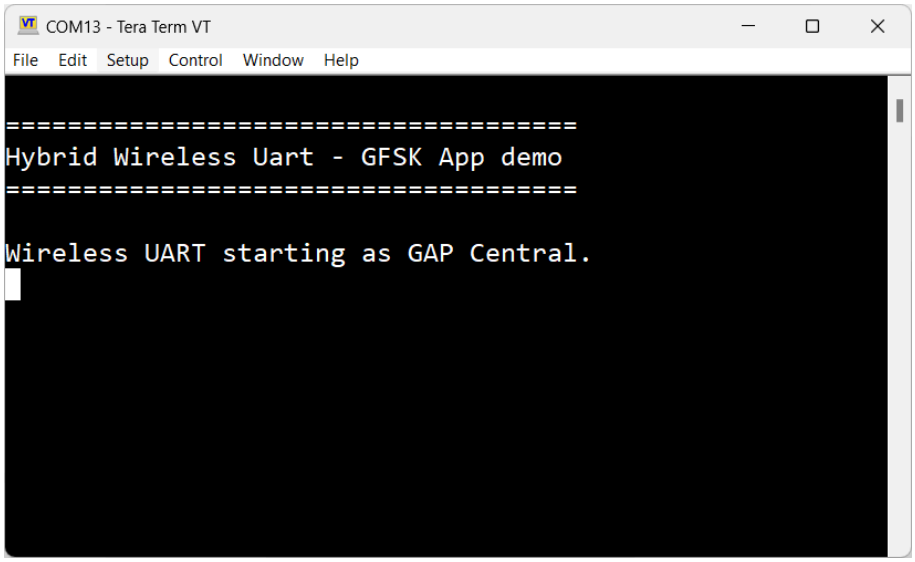

Figure 33. Teraterm – Hybrid Wireless UART (Bluetooth LE) Generic FSK Application start screen

2. The application starts as a GAP central. To switch the role to a GAP peripheral, press the **ROLESW**. Depending on the role, when pressing the **SCANSW** button, the application starts either scanning or advertising.
3. When 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 34.

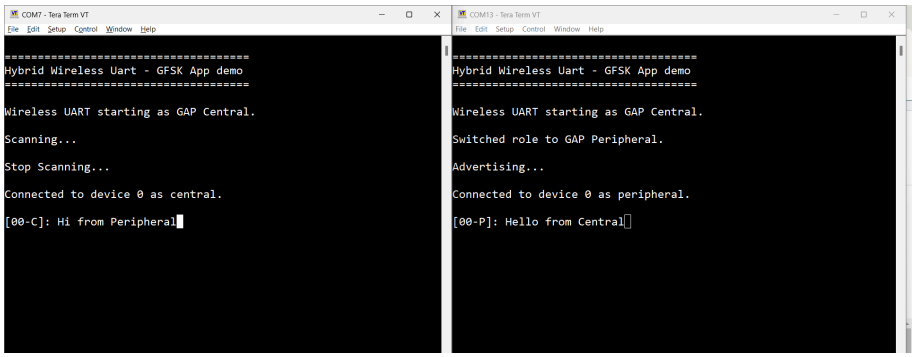

Figure 34. Teraterm – received text on Wireless UART (Bluetooth LE)

Case 2 (Generic FSK): When operating the Generic FSK mode, follow the steps below:

1. Open a serial port terminal and connect them to the two boards. For the procedure, refer to the Section "Testing devices" in *Bluetooth Low Energy Demo Applications User Guide*. The start screen after the board is reset is the same as in Figure 34.

2. The Generic FSK communication direction is not preset. To start receiving, on one board, double click the **SCANSW** button. Then, the device starts receiving packets on the same channel as the Bluetooth LE channel 37.
3. To start transmitting, on another board, long press the **ROLESW** button. The transmitting device uses an identifier known by the receiver and its packets are displayed in the CLI as shown in Figure 35.

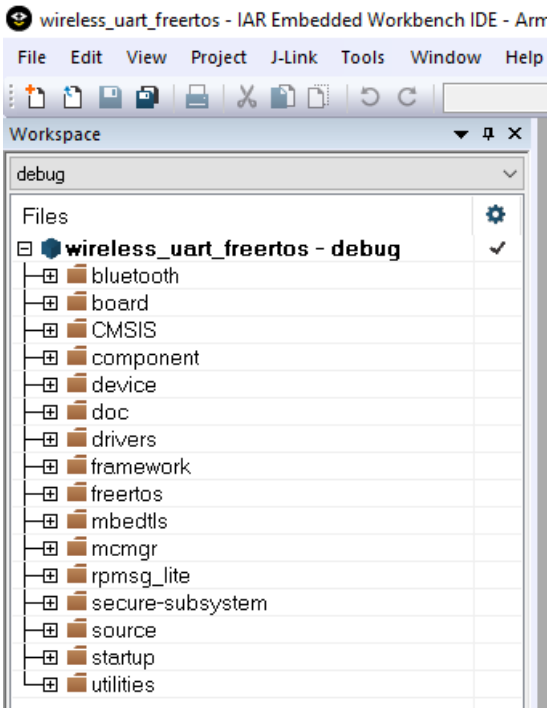

Figure 35. Teraterm – packet transmit/receive on Generic FSK

4. To stop Generic FSK reception, double click the **ROLESW** button.

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

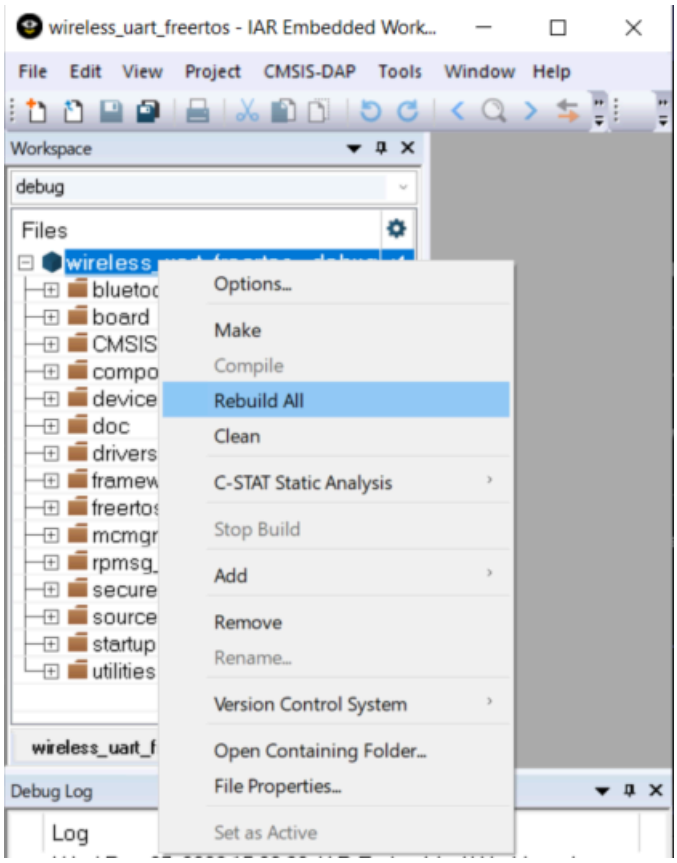

Figure 36. Teraterm – stop Generic FSK activity

Case 3 (Dual-Mode):

The application is ready to combine both scenarios. It can either establish a Bluetooth LE connection and perform Generic FSK, or during Generic FSK activity, the Bluetooth LE connection can be established, and the Controller pauses receiving or announce the discarded Generic FSK transmissions, that are resumed after the Bluetooth LE activity is finished.

1. Establish a Bluetooth LE connection as described in Case 1.
2. Start Generic FSK activity as described in Case 2, independent of the Bluetooth LE roles chosen at Case 1.
3. Start typing in either of the terminals and you can observe the result as shown in Figure 37. The Bluetooth LE activity is prioritized, characters are displayed in the peer's terminal and Generic FSK continues in the available slots, not used by the Bluetooth LE link.

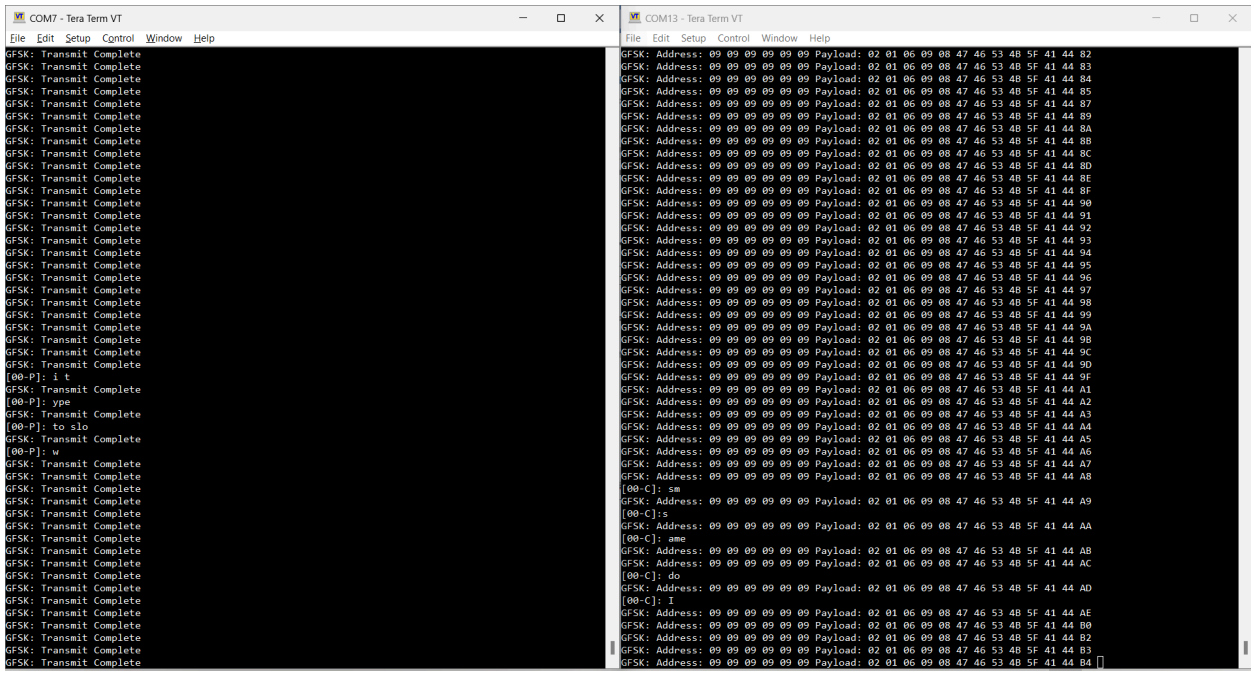

Figure 37. Mixed Wireless UART (Bluetooth LE) and Generic FSK activity

6.5 Customization

Use the steps described in this section for changing the default settings of this demo.

For Bluetooth LE, the default advertising config (`gAdvParams`) are 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, this can be changed.*

For Generic FSK, the following defines of interest are available in `genfsk_app.h`, described in Table 2:

Table 2. Defines

Define	Description
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. Beware, it should also be changed on the receiver in the `hybrid_gfsk.c` controller file.
gGenFSK_H0Value_c	H0 value is used in the header.
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_TxInterval_c	This is the interval for which the transmitter repeats the transmission of a packet. It is set in milliseconds.

6.6 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-dual mode, described in Table 3 below.

Table 3. File Description

File Name	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.

7 References

For more information, refer to NXP website or contact your local Field Application Engineer (FAE).

8 Acronyms and abbreviations

The following acronyms are used in this document.

Table 4. Acronyms and abbreviations

Acronym	Description
Bluetooth LE	Bluetooth Low Energy
EVK	Evaluation Kit
IDE	Integrated Design Environment
ISP	In-system Programming
IoT	Internet of Things
RTOS	Real-time Operating System
SDK	Software Development Kit
UART	Universal Asynchronous Receiver Transmitter
USB	Universal Serial Bus

9 Note about the source code in the document

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

Copyright 2023-2024 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.

10 Revision history

Table 5 summarizes the revisions to this document.

Table 5. Revision history

Document ID	Release date	Description
UG10181 v.1.0	26 November 2024	Initial release for KW47 EAR 2.1 milestone

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.

Contents