AN13953

Integrating NFC Reader Library in a KW4x Bluetooth Low Energy Application

Rev. 1 — 30 May 2023 Application note

Document information

Information	Content
Keywords	AN13953, Bluetooth LE, KW45/KW47, NFC, Reader library, PN5180, CLRC663, NCF3320
Abstract	This document gives instructions on how to create a Bluetooth Low Energy project for the EVK-KW45 development board and MCUXpresso IDE, and how to integrate NFC Reader Library.

1 Introduction

This document gives instructions on how to create a Bluetooth Low Energy (Bluetooth LE) project for the EVK-KW45 development board and MCUXpresso IDE, and how to integrate NFC Reader Library.

This document guides software developers who want to use, adapt, and integrate the NFC Reader Library to an SDK wireless connectivity example.

1.1 NFC reader library overview

The NXP NFC Reader Library is a modular software library written in C language, which provides an API that enables customers to create their own software stack and applications for the NXP contactless reader ICs:

- PN512
- CLRC633 family
- PN7462 family
- PN5180

This API facilitates the most common operations required in NFC applications such as:

- reading or writing data into contactless cards or tags;
- exchanging data with other NFC-enabled devices;
- allowing NFC reader ICs to emulate cards.

The NFC Reader Library is designed in a way to be easily portable to many different microcontrollers with a multi-layered architecture:

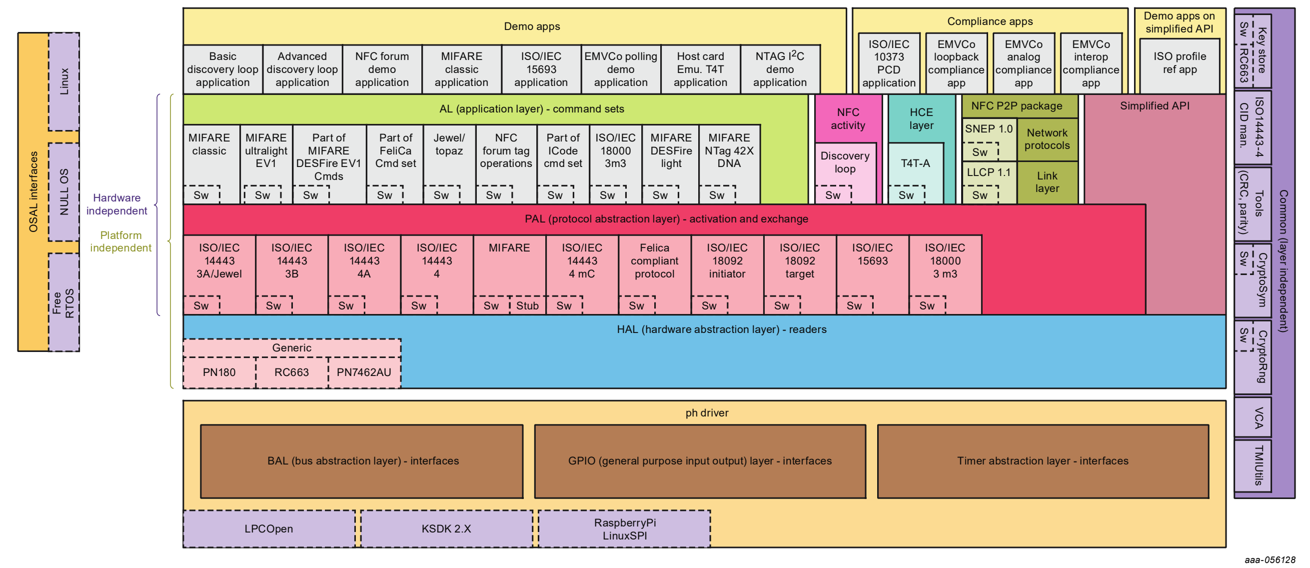


Figure 1. NFC reader library

As main blocks, we have:

- Application Layer (AL) - implements the command sets to interact with MIFARE cards and NFC tags;
- NFC activity - implements a configurable Discovery loop for the detection of contactless cards, NFC tags, or other NFC devices;
- HCE and P2P components, for the emulation of Type 4 tags and P2P data exchange respectively;
- Protocol abstraction layer (PAL) - contains the RF protocol implementation of the ISO14443, FeliCa, vicinity and NFC standards;

- Hardware abstraction layer (HAL) - implements the drivers for controlling the NFC front ends RF interface and capabilities;
- Driver Abstraction Layer (DAL) - implements the GPIO pinning, the timer configuration, and the physical interface (BAL) between the host MCU and the reader IC;
- OSAL module, in charge of abstracting the OS or RTOS specifics (tasks events, semaphores, and threads).

1.2 KW45 wireless microcontroller overview

The KW45 product family is a low-power, highly secure, and single-chip wireless MCU that integrates a high performance Bluetooth Low Energy version 5.3 radio and CAN FD for Automotive and Industrial applications.

The KW45 microcontroller is a low-power, highly secure, Arm Cortex-M33-based wireless device targeting these applications with up to 1 MB program Flash, 128 kB SRAM, and a 2.4 GHz upgradable radio supporting up to 24 simultaneous hardware connections in any central/peripheral combination.

For more details, see Bluetooth® Smart/Bluetooth Low Energy.

1.3 NFC reader library – integration with KW45B41Z-EVK overview

The current NFC Reader Library v5.22.01 does not support Kinetis KW4x MCU. For reference, see the K82 NFC Reader Library package.

To integrate the library, perform the following steps:

- Hardware preparation (the connection between KW45B41Z-EVK board and NFC reader board);
- Setting up the development environment (SDK download, workspace);
- Preparing adaptation files for KW45 board;
- Integrating NFC application to `wireless_uart` Bluetooth LE example;
- Running the demo.

2 Hardware setup

2.1 Hardware required

To perform the NFC integration to a `wireless_uart` Bluetooth LE example, the platforms below are needed:

- NCF3320 Antenna v1.0 board as an NFC transceiver;
- KW45B41Z-EVK board as host MCU, used to load and run the Bluetooth Low Energy Stack and NFC application logic.

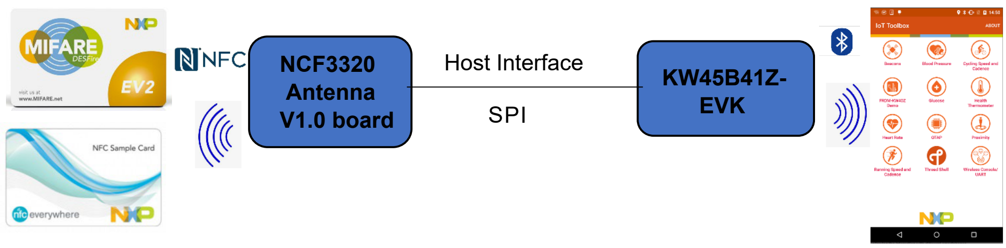


Figure 2. Hardware required for NFC integration

2.2 Pin configuration

The communication between the boards is via the SPI communication using the following pin configuration:

Table 1. Pin configuration

Master board (KW45B41Z-EVK)	Connects to	Slave board (NCF3320 Antenna v 1.0)
PTC0 (J2-pin10)	—	IRQ
PTC1 (J2-pin9)	—	Reset
PTA18 (J2-pin2)	—	MOSI
PTA17 (J1-pin2)	—	MISO
PTA19 (J1-pin5)	—	SCK
PTA16 (J1-pin1)	—	CS
GND (J3-pin7)	—	GND

2.3 Power configuration

The KW45 must be put in Bypass mode by placing the jumper JP5 in position 3-4 (its default position is 1-2, DCDC mode).

	P3V3 JP5	VDD_CORE JP15 [0.84 V - 1.21 V]	VDD_PA JP11	VDD_BRD JP10
DCDC mode	3 - 4	NC	Open	5 - 6
Bypass mode	3 - 4	NC	Open	4 - 3
PMIC mode	5 - 6	1-2	Ext supply	1 - 2

aaa-056129

Figure 3. Power configuration

3 Setting up the development environment

Follow the next steps to create an EVK-KW45 Bluetooth Low Energy project for MCUXpresso IDE (in which the NFC library is integrated) and to get the latest NFC Reader Library release. For this example, the version of SDK used is **SDK_2_12_2_KW45B41Z-EVK** and the **NFC Reader Library** version is **v5.22.01**.

1. Download and Install MCUXpresso IDE (for this example, we are using **MCUXpresso IDE v11.7.0 [Build 9198**]).
   - Go to the MCUXpresso-IDE webpage and download the latest version of IDE.

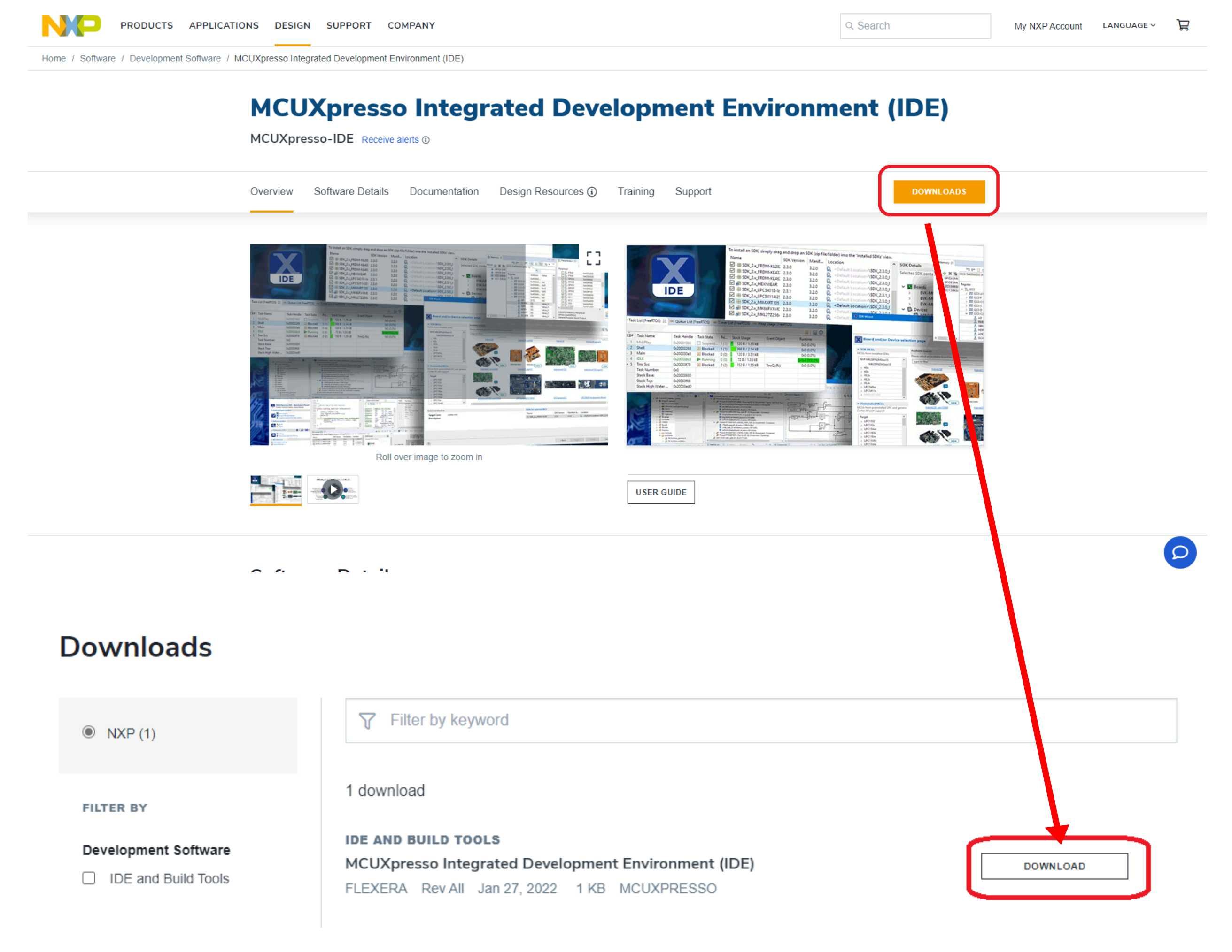


Figure 4. Download IDE

- Install the IDE.

2. Get the latest NFC Reader Library release (for this example, we are using v5.22.01)
   - Go to the NXP NFC Reader Library webpage.
   - Go to the **Downloads** tab.

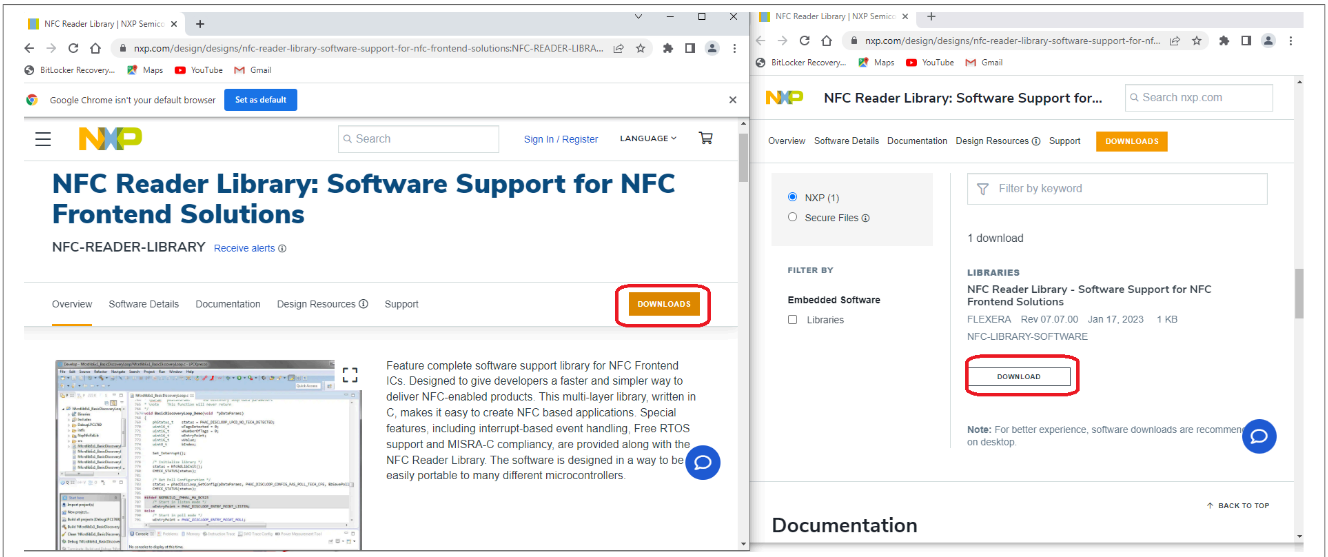
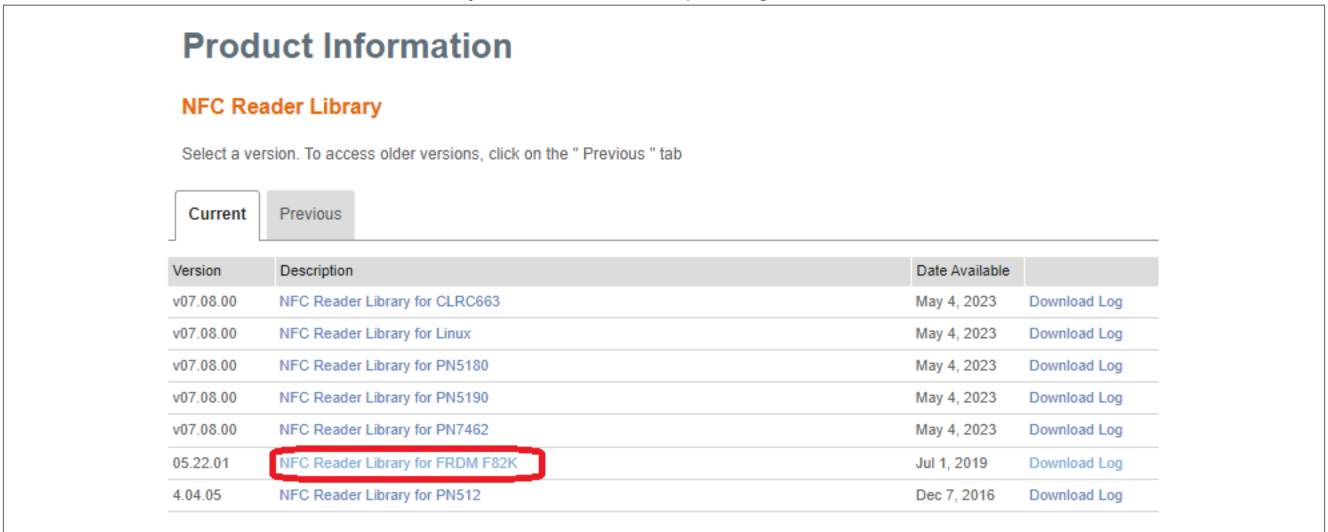


Figure 5. Downloads tab

- Download the NFC Reader Library for Kinetis K82F package.



Product Information

NFC Reader Library

Select a version. To access older versions, click on the " Previous " tab

Current | Previous

Version	Description	Date Available	
v07.08.00	NFC Reader Library for CLRC663	May 4, 2023	Download Log
v07.08.00	NFC Reader Library for Linux	May 4, 2023	Download Log
v07.08.00	NFC Reader Library for PN5180	May 4, 2023	Download Log
v07.08.00	NFC Reader Library for PN5190	May 4, 2023	Download Log
v07.08.00	NFC Reader Library for PN7462	May 4, 2023	Download Log
05.22.01	NFC Reader Library for FRDM F82K	Jul 1, 2019	Download Log
4.04.05	NFC Reader Library for PN512	Dec 7, 2016	Download Log

Figure 6. Download NFC reader library

3. Generate a downloadable SDK package for KW45B41Z-EVK board (**SDK_2_12_2_KW45B41Z-EVK**).
   - Navigate to Select Development Board webpage and search for KW45B41Z-EVK board.
   - Select **Build MCUXpresso SDK**.

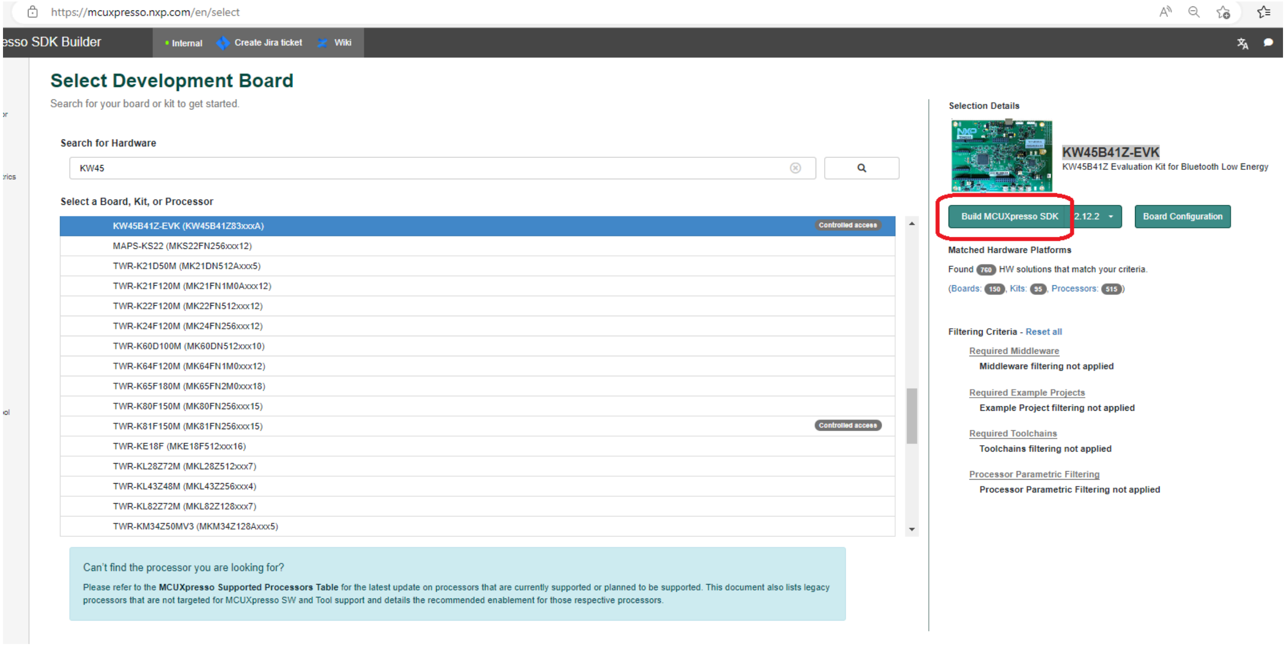


Figure 7. Select Development Board page

– As a toolchain, make sure that the MCUXpresso IDE is selected.

– Use the **Download SDK** button to start downloading SDK package.

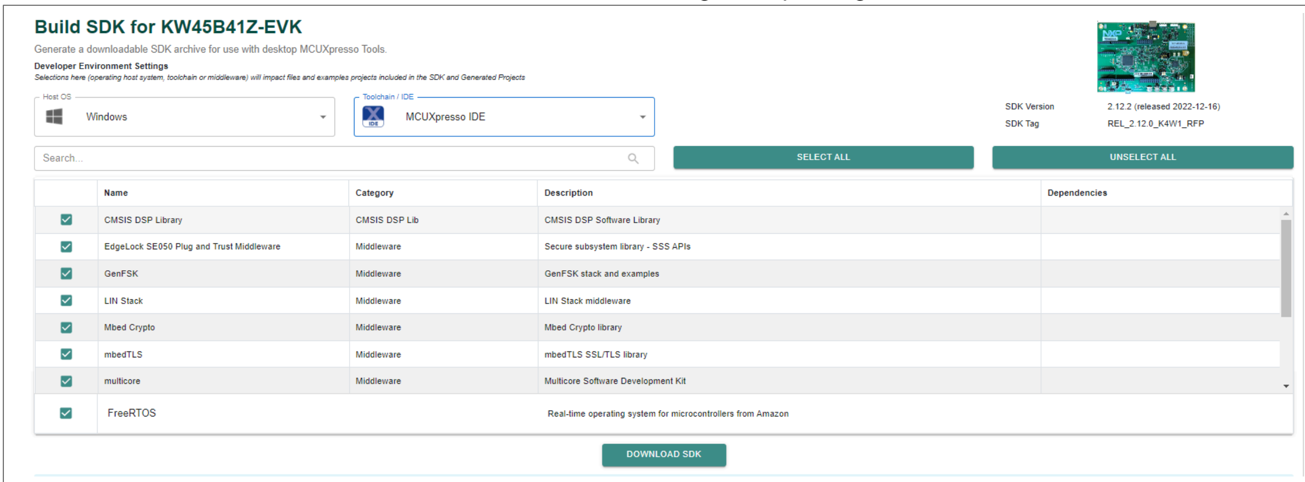


Figure 8. Start downloading

4. Create the MCUXpresso workspace.
   - Open MCUXpresso IDE and create a workspace.
   - Drag and drop **SDK_2_12_2_KW45B41Z-EVK** into the installed SDKs tab of the MCUXpresso IDE.

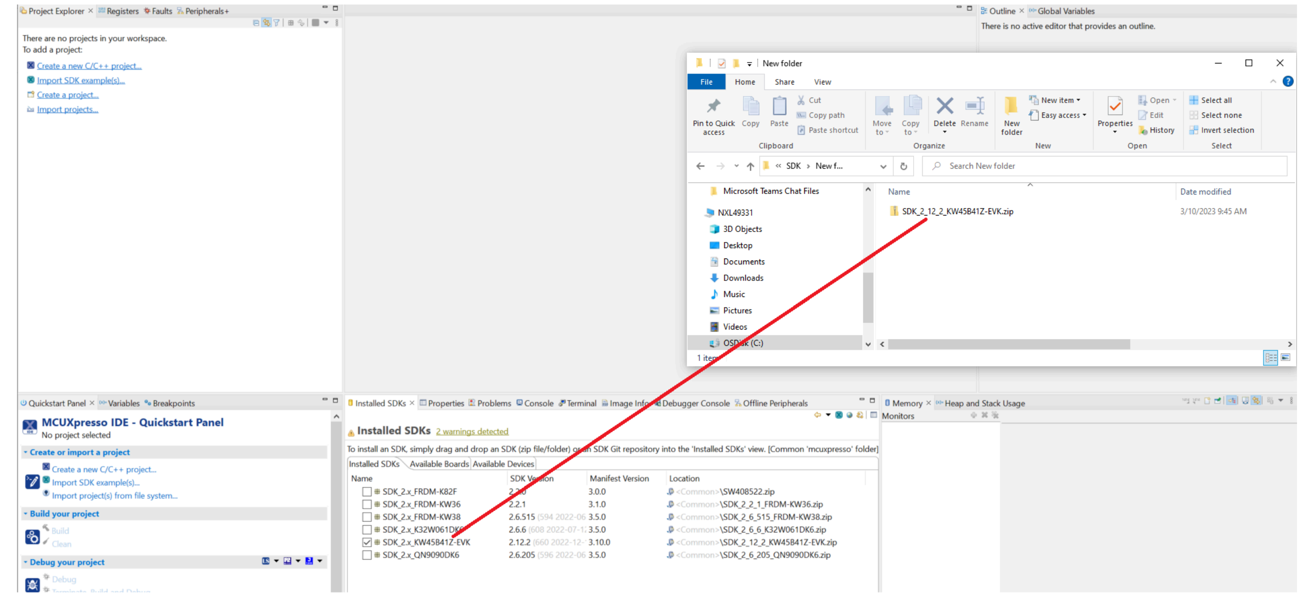


Figure 9. Dragging and dropping `SDK_2_12_2_KW45B41Z-EVK`

- Click **Import SDK example(s)** from the **Quickstart** panel.
- Search and select **kw45b41zevk** in the **board and/or device selection** page. Then, click **next**.

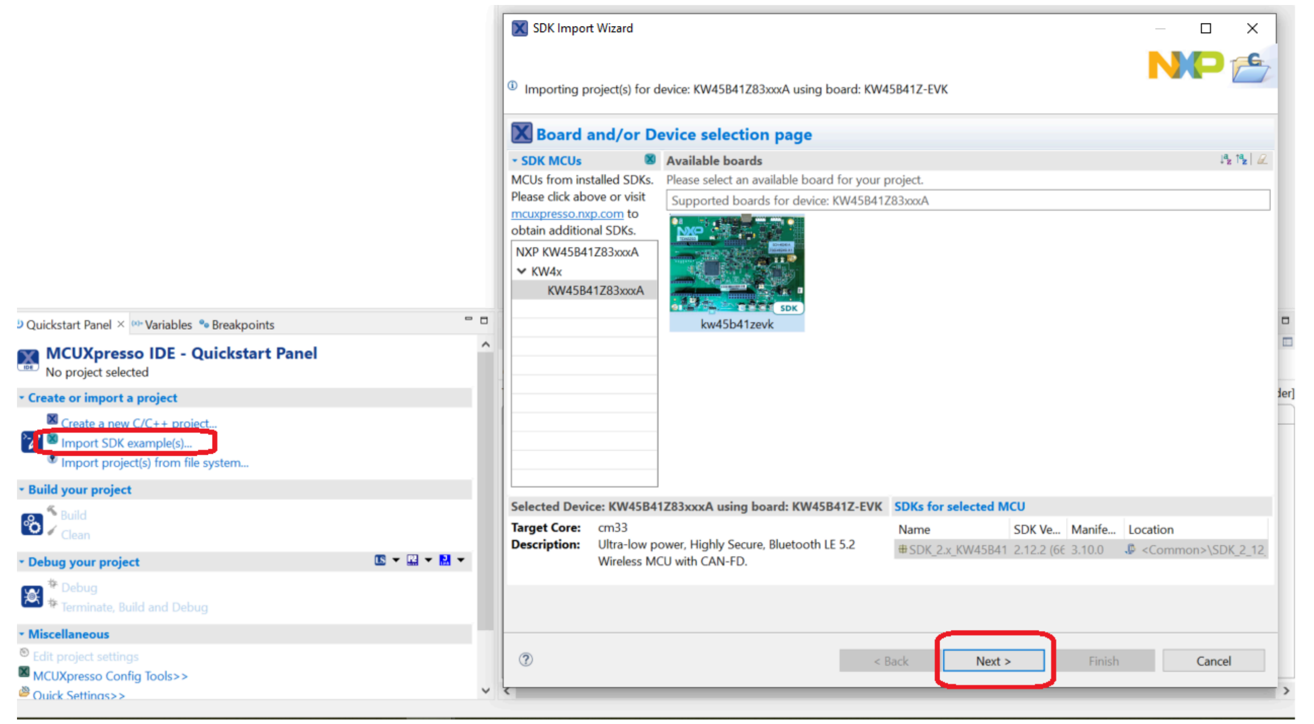


Figure 10. Select `kw45b41zevk`

- Select **wireless_uart_freertos** Bluetooth LE project。

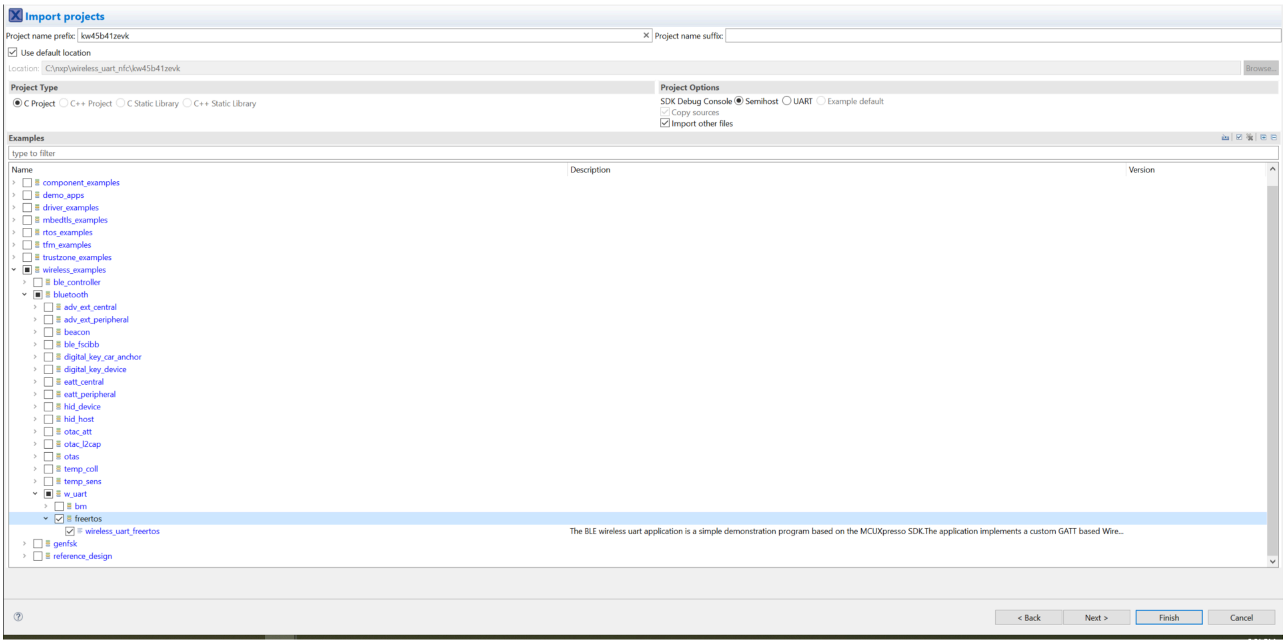


Figure 11. Selecting `wireless_uart_freertos`

4 Preparing adaptation files for KW45B41Z-EVK board

After unzipping the NFC Reader Library, create an equivalent file for KW45B41Z-EVK (*Board_FRDM_KW45FRc663.h*) by setting the right configuration for GPIOs and handlers.

The following files below must also be adapted:

- `DAL\cfg\BoardSelection.h`: include the new header file created.
- `DAL\src\KinetisSDK\phbalReg_KinetisSpi`: configure low-power SPI, include the LPIT driver, and add the SPI initialization afferent to K45 as follows:

```
        lpspi_master_config_t g_masterConfig;

     if ( (pDataParams == NULL) || (sizeof(phbalReg_Type_t) !=
  wSizeOfDataParams))
     {
         return (PH_DRIVER_ERROR | PH_COMP_DRIVER);
     }
     ((phbalReg_Type_t *)pDataParams)->wId      = PH_COMP_DRIVER |
  PHBAL_REG_KINETIS_SPI_ID;
     ((phbalReg_Type_t *)pDataParams)->bBalType = PHBAL_REG_TYPE_SPI;
     memset(&g_masterConfig, 0, sizeof(lpspi_master_config_t));
     g_masterConfig.baudRate = PHDRIVER_KSDK_SPI_DATA_RATE;
     g_masterConfig.bitsPerFrame = 8U;
     g_masterConfig.cpol = kLPSPI_ClockPolarityActiveHigh;
     g_masterConfig.cpha = kLPSPI_ClockPhaseFirstEdge;
     g_masterConfig.direction = kLPSPI_MsbFirst;
     g_masterConfig.pcsToSckDelayInNanoSec = 1000000000U /
  PHDRIVER_KSDK_SPI_DATA_RATE;
     g_masterConfig.lastSckToPcsDelayInNanoSec = 1000000000U /
  PHDRIVER_KSDK_SPI_DATA_RATE;
     g_masterConfig.betweenTransferDelayInNanoSec = 1000000000U /
  PHDRIVER_KSDK_SPI_DATA_RATE;
     g_masterConfig.whichPcs = kLPSPI_Pcs0;
```

```
    g_masterConfig.pcsActiveHighOrLow = kLPSPI_PcsActiveLow;

    /*Set clock source for LPSPI and get master clock source*/
   CLOCK_SetIpSrc(kCLOCK_Lpspi0, kCLOCK_IpSrcFro192M);
   CLOCK_SetIpSrcDiv(kCLOCK_Lpspi0, kSCG_SysClkDivBy16);
   phbalReg_SpiInit();
    /* Initialize the DSPI peripheral */
    LPSPI_MasterInit(PHDRIVER_KSDK_SPI_MASTER, &g_masterConfig,
 CLOCK_GetIpFreq(PHDRIVER_KSDK_SPI_CLK_SRC));
```

- `DAL\src\KinetisSDK\phDriver_KinetisSDK.c`: include the `lpit` driver, update interrupt types, and initialize the `lpit` timer as follows:

```

    lpit_chnl_params_t lpitChannelConfig;
    lpitChannelConfig.chainChannel          = false;
    lpitChannelConfig.enableReloadOnTrigger = false;
    lpitChannelConfig.enableStartOnTrigger  = false;
    lpitChannelConfig.enableStopOnTimeout   = false;
    lpitChannelConfig.timerMode             = kLPIT_PeriodicCounter;
    /* Set default values for the trigger source */
    lpitChannelConfig.triggerSelect = kLPIT_Trigger_TimerChn0;
    lpitChannelConfig.triggerSource = kLPIT_TriggerSource_External;

    LPIT_SetupChannel(PH_DRIVER_KSDK_PIT_TIMER,
 PH_DRIVER_KSDK_TIMER_CHANNEL,&lpitChannelConfig);
    LPIT_StopTimer(PH_DRIVER_KSDK_PIT_TIMER, PH_DRIVER_KSDK_TIMER_CHANNEL);
    LPIT_ClearStatusFlags(PH_DRIVER_KSDK_PIT_TIMER, kLPIT_Channel0TimerFlag);
    LPIT_EnableInterrupts(PH_DRIVER_KSDK_PIT_TIMER,
 kLPIT_Channel0TimerInterruptEnable);

    /* Configure timer period */
    LPIT_SetTimerPeriod(PH_DRIVER_KSDK_PIT_TIMER, PH_DRIVER_KSDK_TIMER_CHANNEL,
 (uint32_t)qwTimerCnt);
```

Also in this source an equivalent function for PIT IRQ Handler is needed:

```
void PIT_DriverIRQHandler(void)
{
    /* Clear interrupt flag.*/
    LPIT_ClearStatusFlags(PH_DRIVER_KSDK_PIT_TIMER, kLPIT_Channel0TimerFlag);

    /* Single shot timer. Stop it. */
    LPIT_StopTimer(PH_DRIVER_KSDK_PIT_TIMER, PH_DRIVER_KSDK_TIMER_CHANNEL);
    LPIT_DisableInterrupts(PH_DRIVER_KSDK_PIT_TIMER,
 kLPIT_Channel0TimerInterruptEnable);

    pPitTimerCallBack();

}
```

- In *NxpNfcRdLib\types\ph_NxpBuild_Platform.h*, add a definition for KW45 board.

For additional changes that are not outlined in this chapter, see the source code.

5 Integrating NFC application to `Wireless_UART` Bluetooth LE example

This chapter introduces how to integrate the `BasicDiscoveryLoop` NFC example to `Wireless_UART` Bluetooth LE application.

For the integration, perform the following steps:

- Add `pit` drivers in the project.
- On the `wireless_uart` project location, create an **nfc** folder.

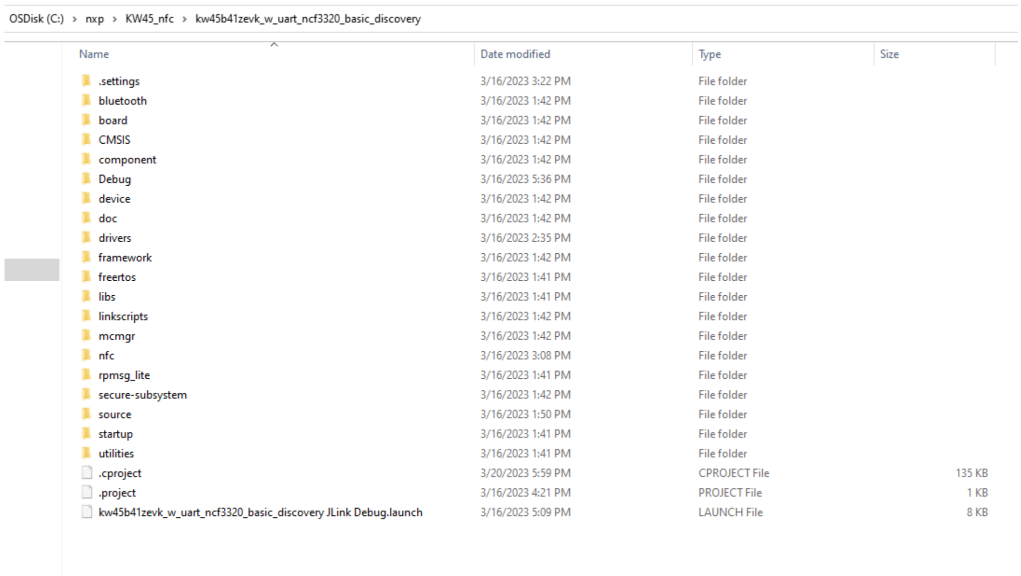


Figure 12. Creating an *nfc* folder

- Copy the *DAL*, *NxpNfcRdLib*, and *phOsal* folders from the modified NFC Reader Library.

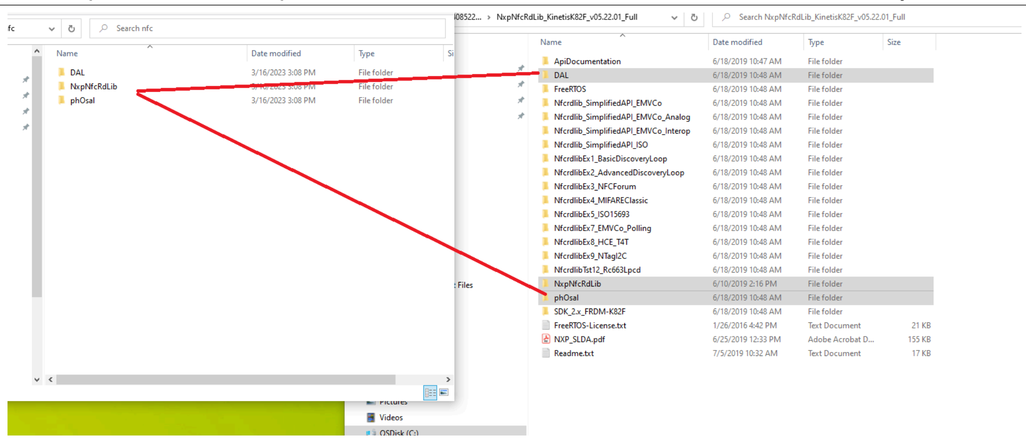


Figure 13. Copy folders

- On the `wireless_uart` project location, the *source* folder, create an *nfc* subfolder to integrate the `BasicDiscovery` loop files from *NfcrdlibEx1_BasicDiscoveryLoop* folder.

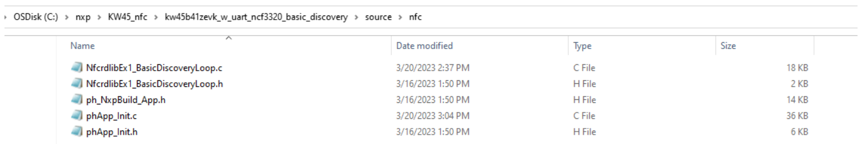


Figure 14. Source folder

These files require the changes below:

- In *NfcrdlibEx1_BasicDiscoveryLoop.c* file, rename main function to `NFC_BasicDiscoveryLoop_Start` and remove OS initialization.
- In *NfcrdlibEx1_BasicDiscoveryLoop.h*, declare `NFC_BasicDiscoveryLoop_Start` function.
- In *ph_NxpBuild_App.h*, add the definition for KW45,

– In *phApp_Init.h*, add the definition for `ph` driver.
– In *phApp_Init.c*, include the needed headers add the CPU initialization for KW45 in *phApp_CPU_Init* function and add the IRQ handler as follows:

```
IRQ_HANDLE_ARRAY_DEFINE(g_IrqHandle, 1);
#define gBoardIrqHandle ((irq_handle_t)g_IrqHandle[0])

void CLIF_HalIRQHandler(void *param)
{
    CLIF_IRQHandler();
}
```

Configure the GPIO pins in the `phApp_Configure_IRQ` function:

```
#if !gAppButtonCnt_c
    NVIC_SetPriority(EINT_IRQn, EINT_PRIORITY);
    NVIC_ClearPendingIRQ(EINT_IRQn);
    EnableIRQ(EINT_IRQn);
#else
#include "fsl_adapter_gpio.h"
    static const hal_gpio_pin_config_t g_IrqConfig = {
    kHAL_GpioDirectionIn,
    PHDRIVER_IRQ_GPIO_PIN_DEFAULT_STATE,
    PHDRIVER_IRQ_GPIO_PORT_INSTANCE,
    PHDRIVER_IRQ_GPIO_PIN
    };
        if(kStatus_HAL_GpioSuccess !
=HAL_GpioInit((hal_gpio_handle_t)gBoardIrqHandle,

 (hal_gpio_pin_config_t *)&g_IrqConfig))
    {
        return PH_ERR_ABORTED;
    }
    (void)HAL_GpioInstallCallback(gBoardIrqHandle,CLIF_HalIRQHandler, NULL);
```

Also, call `BleApp_SendNfcCardInfo` in `phApp_PrintTagInfo` function.
– In *wireless_uart.h*, define the `nfcCardInfo_tag` structure, declare `BleApp_SendNfcCardInfo` function, and define a `HexToAscii` function:

```
typedef struct nfcCardInfo_tag
{
 uint8_t technology;
 uint8_t uid[8];
}nfcCardInfo_t;

extern nfcCardInfo_t gNfcCardInfo;

void BleApp_SendNfcCardInfo(void);
#define HexToAscii(hex) (uint8_t)( ((hex) & 0x0F) + ((((hex) & 0x0F) <= 9) ?
 '0' : ('A'-10)) )
```

– In *wireless_uart.c*, call the initializations for `pit` module and the `nfc` basic discovery start and add definitions for `BleApp_SendNfcCardInfoHandler` and `BleApp_SendNfcCardInfo` functions.

```
static void BleApp_SendNfcCardInfoHandler(void *pParam)
{
 static uint8_t dataToSend = 1;
 uint8_t *pString;
 uint8_t stringSize;
 uint8_t index = 0;
```

```
 uint8_t hexString1[13] = {"Card Detected"};
 uint8_t hexString2[12] = {"Technology=A"};
 uint8_t hexString3[20] = {"UID=0000000000000000"};

 if(dataToSend == 1)
 {
  pString = hexString1;
  stringSize = 13;
  dataToSend = 2;
 }
 else if (dataToSend == 2)
 {
  hexString2[11] = gNfcCardInfo.technology;
  pString = hexString2;
  stringSize = 12;
         dataToSend = 3;
 }
 else
 {
  index = 4;
  for(uint8_t i = 0; i < 8; i++)
  {
   hexString3[index++] = HexToAscii( (gNfcCardInfo.uid[i])>>4 );
   hexString3[index++] = HexToAscii( gNfcCardInfo.uid[i] );
  }
  pString = hexString3;
  stringSize = 20;
  dataToSend = 1;
 }

 BleApp_SendUartStream(pString, stringSize);

 if(dataToSend!=1)
 {

 (void)App_PostCallbackMessage((appCallbackHandler_t)BleApp_SendNfcCardInfo,
 NULL);
 }
}

void BleApp_SendNfcCardInfo(void)
{
    (void)App_PostCallbackMessage(BleApp_SendNfcCardInfoHandler, NULL);
}
```

– In *main.c*, include board.

– In *board.c*, include *fsl_lpit.h* and initialize `pit` module.

```
void BOARD_InitPitModule(void)
{
 lpit_config_t pitConfig;         /* Structure of initialize PIT */
           CLOCK_SetIpSrc(kCLOCK_Lpit0, kCLOCK_IpSrcFro6M);
 LPIT_GetDefaultConfig(&pitConfig);
 LPIT_Init(LPIT0, &pitConfig);
}
```

– In **board.h**, declare pit initialization.

- Update the linker information (**Project Properties** -> **C/C++ Build** -> **Settings**) and preprocessor defines (**Project Properties** -> **C/C++ Build** -> **Preprocessor**).

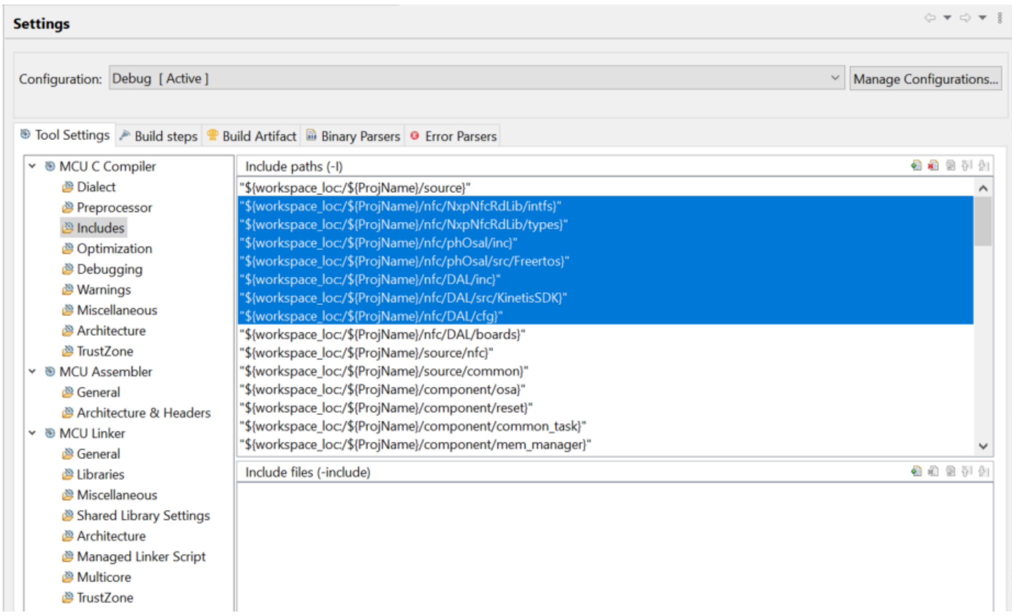


Figure 15. Updating linker information 1

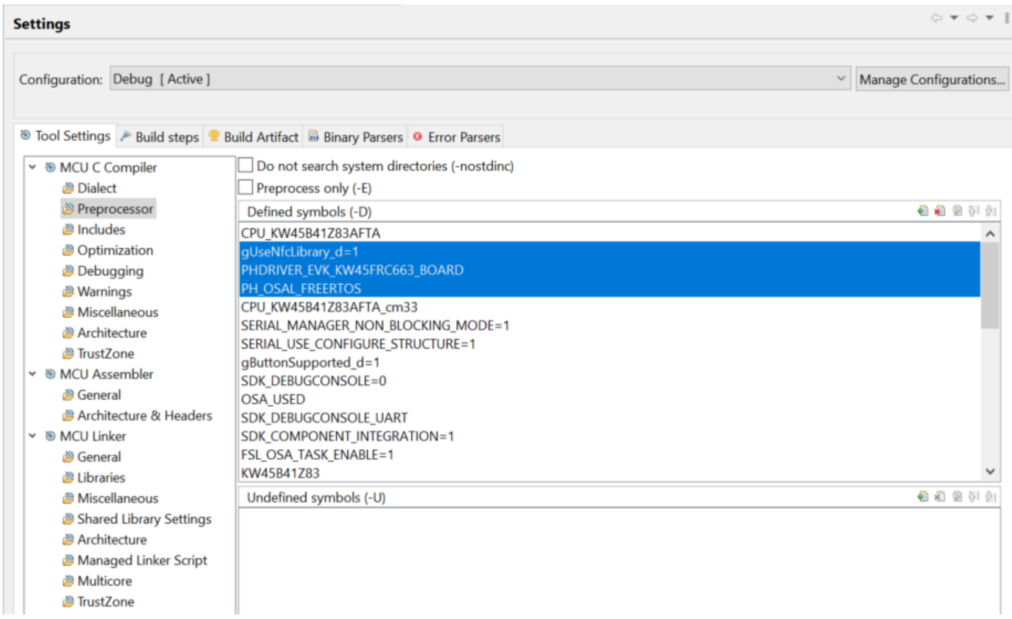


Figure 16. Updating linker information 2

6 Running the demo

To run the demo, perform the following steps:

- Create hardware connection based on Section 2.
- Open a serial terminal on the corresponding COM port for KW45 board. The baud rate used is 115200.
- Start advertising.
- Open **Mobile APP** - **IoT toolbox** - **Wireless UART**. The KW45 board is listed as **NXP_WU**.

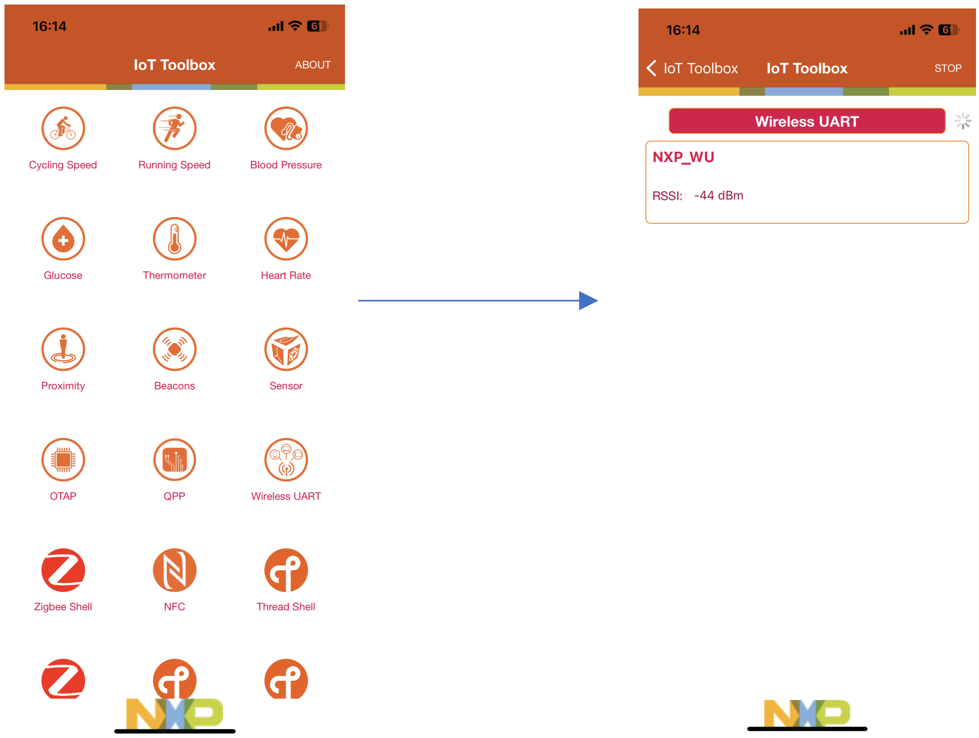


Figure 17.

- Create Bluetooth LE connection. The serial log contains the log for Bluetooth LE operations.

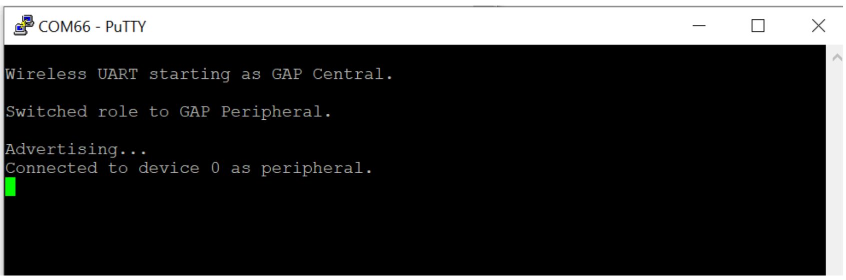


Figure 18. Serial log

- To initiate discovery demo, use NFC cards close to NCF3320 Antenna v1.0 board.
- Once the card is detected, an event is sent to the mobile application including technology and UUID of the card.

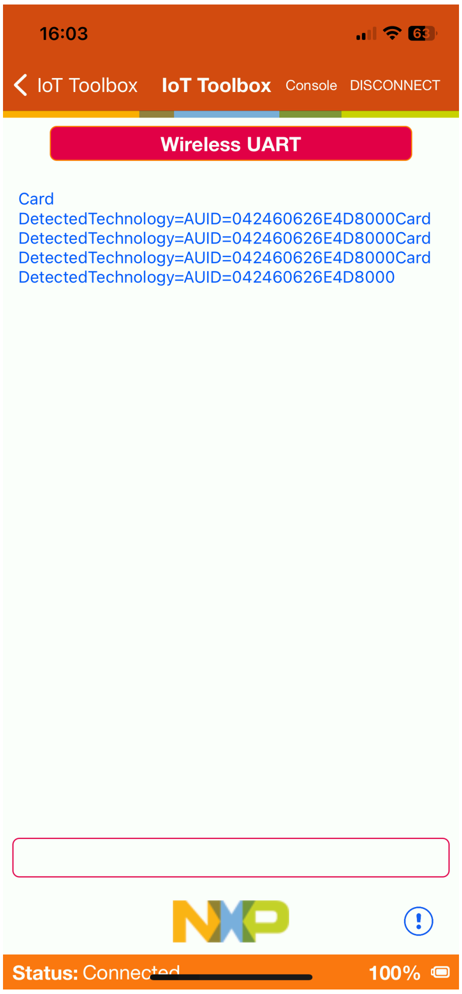


Figure 19.

7 Reference

- NFC Reader Library
- NCF3320
- CLRC663 plus
- FRDM- KW41Z board
- KW45B41Z-EVK SDK
- MCUXpresso IDE

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 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 must be 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 2 summarizes the revisions to this document.

Table 2. Revision history

Document ID	Release date	Description
AN13953 v.1	30 May 2024	Initial public release

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.

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

1	**Introduction**	**2**
1.1	NFC reader library overview	2
1.2	KW45 wireless microcontroller overview	3
1.3	NFC reader library – integration with KW45B41Z-EVK overview	3
2	**Hardware setup**	**3**
2.1	Hardware required	3
2.2	Pin configuration	3
2.3	Power configuration	4
3	**Setting up the development environment**	**4**
4	**Preparing adaptation files for KW45B41Z-EVK board**	**9**
5	**Integrating NFC application to Wireless_ UART Bluetooth LE example**	**10**
6	**Running the demo**	**14**
7	**Reference**	**16**
8	**Note about the source code in the document**	**17**
9	**Revision history**	**17**
	Legal information	**18**

Date of release: 30 May 2023
Document identifier: AN13953