UM10950

Start-up Guide for FRDM-KW41Z Evaluation Board Bluetooth Paring example with NTAG I²C *plus*

Rev. 2.0 — 5 June 2018
422220

User manual
COMPANY PUBLIC

Document information

Info	Content
Keywords	NTAG I²C *plus*, FRDM-KW41Z
Abstract	This document gives a start-up guide for Bluetooth BLE pairing demonstration between FRDM-KW41 and NFC mobile device with use of NTAG I²C *plus* and also explains how the pairing part is included in the connectivity stack.

Revision history

Rev	Date	Description
2.0	20180605	The BT pairing was updated with a different demo application and migrated from Kinetis Design Studio to MCUXpresso IDE and SDK.
1.2	20180206	Change of the information on how to use the demo in chapter 2
1.1	20170307	Text: LPCXpresso IDE changed to Kinetis Design Studio IDE
1.0	20170307	Initial version

Contact information

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

1. Introduction and HW setup

This document can be divided into three basic parts. This the first one introduces and describes the HW setting. The second part is a quick startup guide and describes how to quickly and easily work with the prepared sample of the Bluetooth demo application. The third part of the document describes how to import NTAG I2C middleware with a NDEF library into the selected Bluetooth application.

The whole integration process of the NTAG I²C middleware and the NDEF library can be applied to any customer application.

1.1 HW setup

For the NTAG I²C *plus* development kit, there are 2 kits that can be attached to the FRDM-KW41Z board. The first OM5569 [2] is the original and it represents the typical HW design, the second OM23221ARD [3] is adapted to Arduino pinout. Both kits are fully fledged and its on the user which is chosen.

1.1.1 NTAG I²C *plus* board

The HW connection and wiring between the NTAG I²C *plus* board and FRDM-KW41Z board is shown at the following picture. RGB LED and micro switches SW3, SW4 which are used in HID_device demo application (more information is in the chapter 2.2) and which signal usage of the NTAG I²C chip are mounted directly on the FRDM-KW41Z. Setting functions and control functions for this LED is a part of the SDK. From this reason there is not required extra HW setup for these elements (see reference[2]).

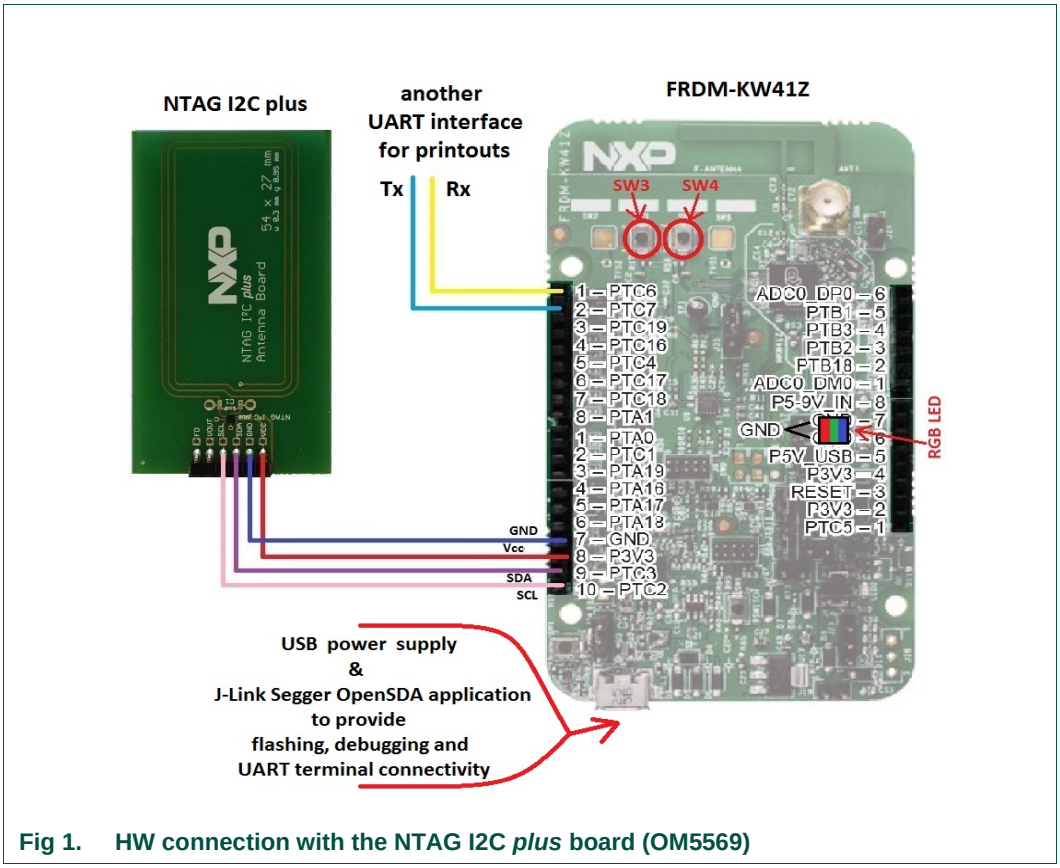

Fig 1. HW connection with the NTAG I2C *plus* board (OM5569)

Table 1. HW configuration within the integration of the NTAG I²C plus PCB board

Example	Description	GPIO	GPIO direction	specification
I2C – SCL	Interface clock	PTC 2	I²C specific	open drain SCL
I2C – SDA	Interface data	PTC 3	I²C specific	open drain SCL
GND	Ground	GND	output	-
VCC_SW	NTAG I²C Antenna board power supply	P3V3	output	-
LED – RED	FRDM-KW41Z - RGB LED driver	PTC 1	output	Default SDK configuration
LED – GREEN	FRDM-KW41Z RGB LED driver	PTA 19	output	Default SDK configuration
LED – BLUE	FRDM-KW41Z RGB LED driver	PTA 18	output	Default SDK configuration
SW3		PTC 4	input	Default SDK configuration
SW4		PTC 5	input	Default SDK configuration

1.1.2 Arduino NTAG I²C *plus* board

OM23221ARD development kit with NTAG I²C *plus* is an easy add-on to many popular MCU boards. It gives connectivity to any device with Arduino pinout. The HW setting is the identical with previous description in the chapter 1.1.1 (see reference [3])

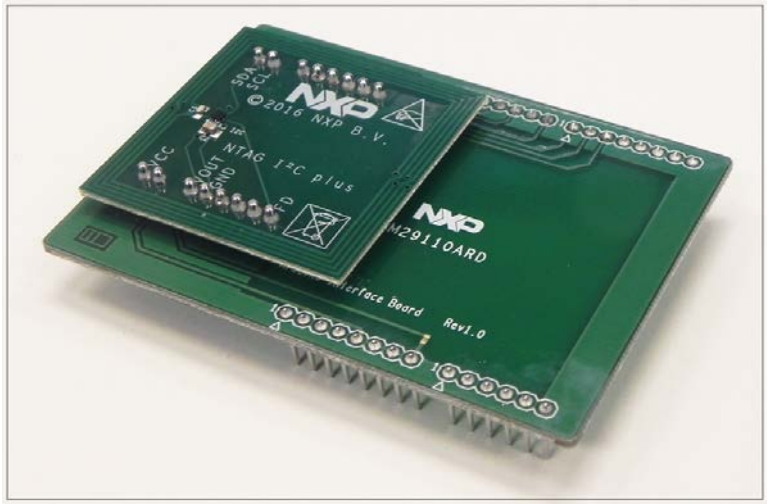

Fig 2. HW connection with the Arduino NTAG I2C *plus* board (OM23221ARD)

2. Quick start-up the demo application

This chapter provides a procedure how to easily install the development environment with the necessary components. Next, how to compile and debug a demo application and finally how to verify it's functionality.

2.1 Installation of the MCUXpresso IDE and SDK

The whole project of the demo application is written for the MCUXpresso IDE, so it is necessary to install this development environment. The next step is to generate and import the SDK (Software Development Kit) needed for the application to be compiled and debugged. The procedure to achieve this is described in the following chapters.

2.1.1 How to download the installation file of the MCUXpresso IDE

The following steps show how to easily download the installation file for the MCUXpresso IDE.

1.) Open the welcome page of the MCUXpresso site [4]: https://mcuxpresso.nxp.com/en/welcome

2.) Select the “SOFTWARE AND TOOLS” and press the “Learn More >”

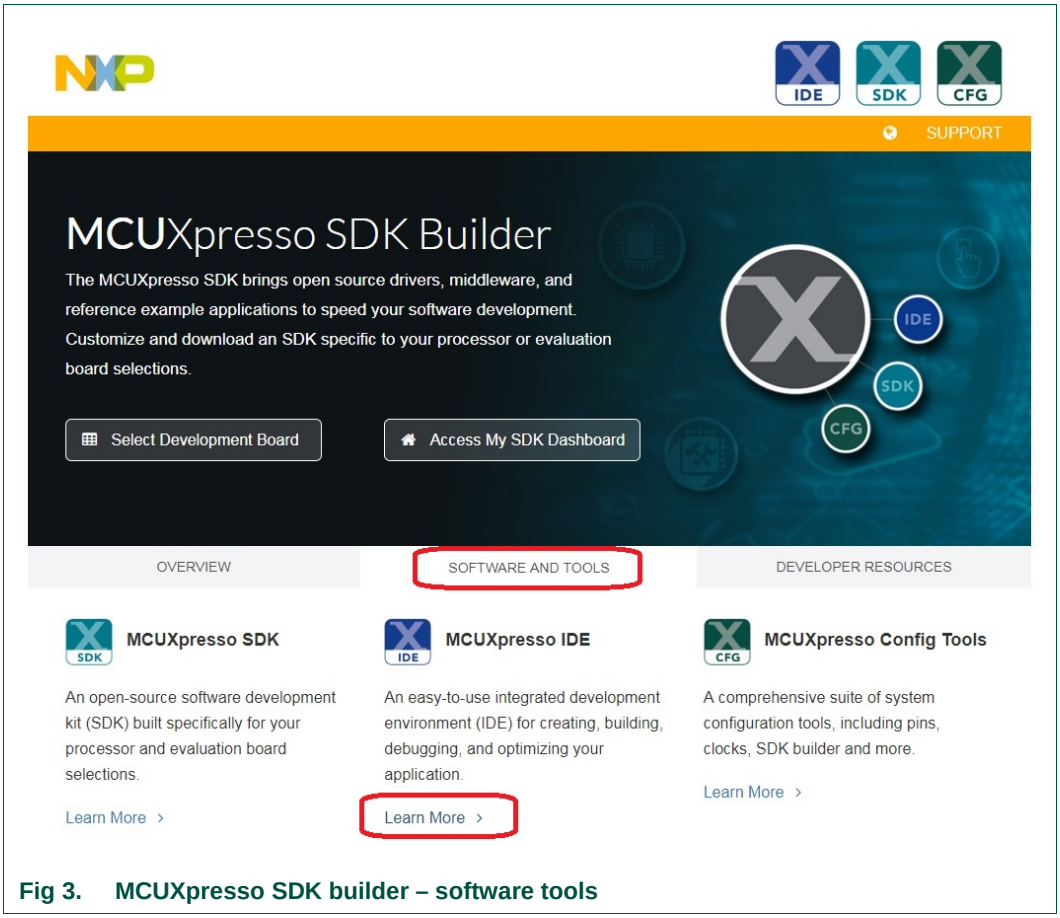

Fig 3. MCUXpresso SDK builder – software tools

3.) In the next step you will be redirected to the NXP web page
4.) Choose the “DOWNLOADS” tab and press the “Download”

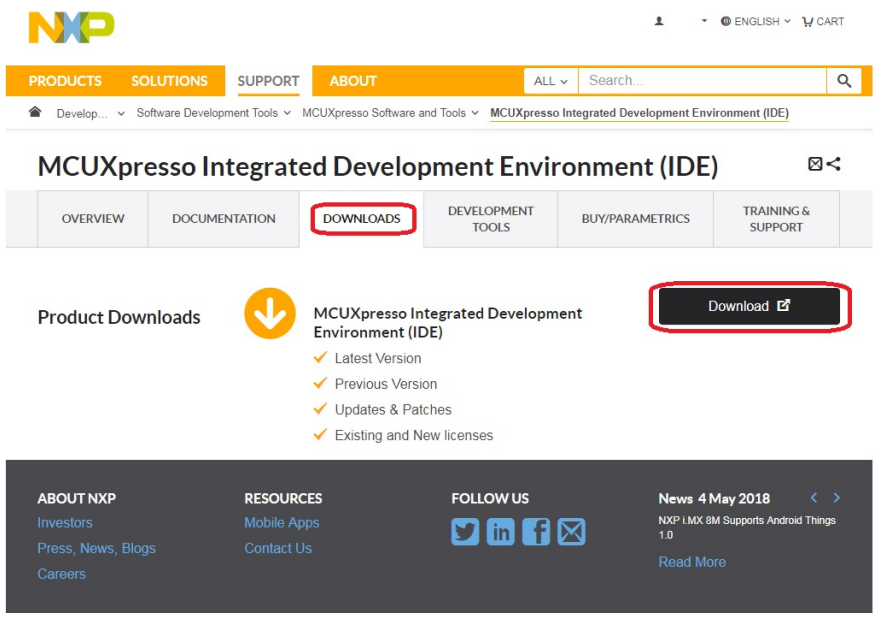

Fig 4. IDE – starting the download

5.) Within the next step is necessary to sign in.
6.) Select the version of the installation file for MCUXpresso. It is recommended to select the latest version (especially at the start of development) because it contains the latest updates.

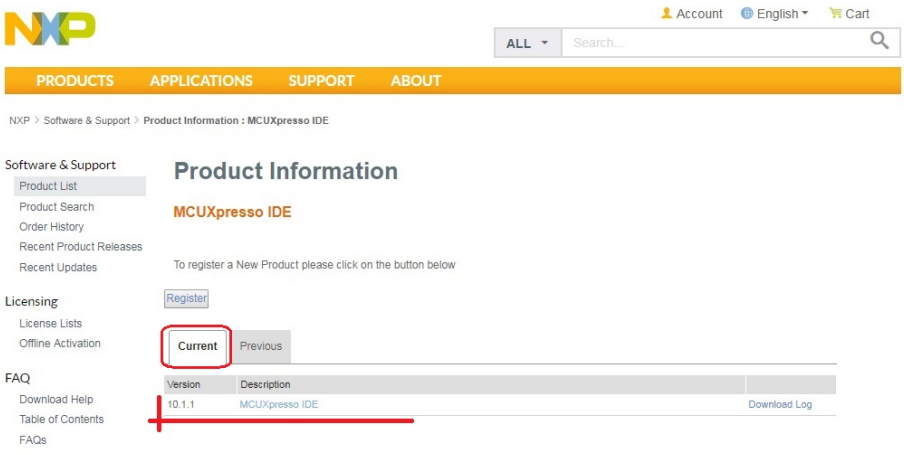

Fig 5. IDE – select a version

7.) Select the correct type of the operating system that is used by user and download the installation file.

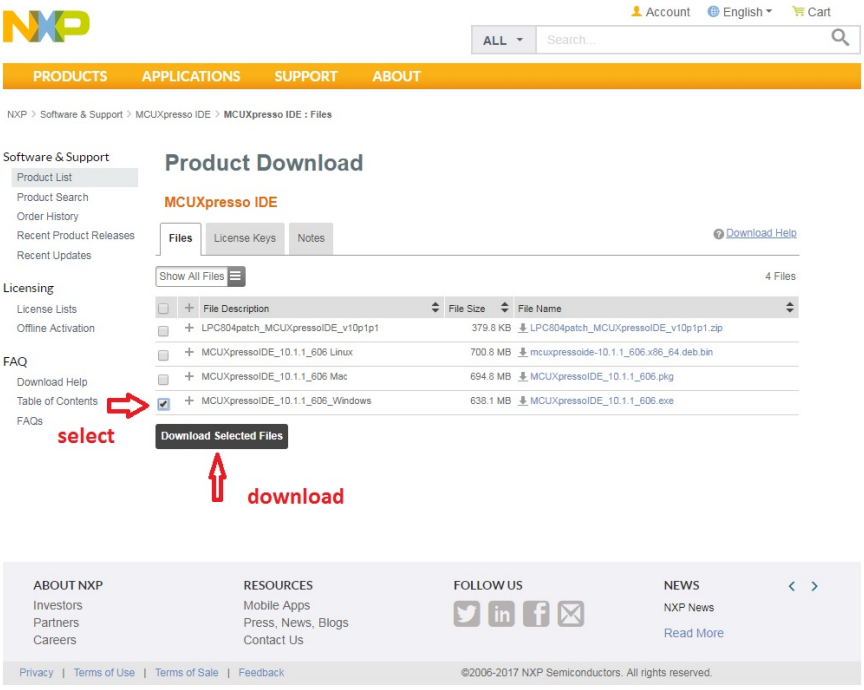

Fig 6. IDE – select and download

2.1.2 Installation of the MCUXpresso IDE

The installation file was obtained using the procedure in the previous chapter 2.1.1. Once it is started, it is recommended that user follows the standard recommended installation setup. Custom installation requires a deeper knowledge of the MCUXpresso IDE.

No additional licenses are required after installing MCUXpresso. The development environment can be used immediately in its entirety.

2.1.3 How to generate MCUXpresso SDK with Bluetooth and NTAG I2C software

From the reason the *ntag_i2c_plus* middleware (which covers the NTAG I²C chip functionality) is missing in the general SDK here is the procedure how to generate the SDK based on the necessary parts for the Bluetooth demo applications and *ntag_i2c_plus* middleware software. Follow the steps bellow.

1.) Open the welcome page of the MCUXpresso site [4]: https://mcuxpresso.nxp.com/en/welcome
2.) Select the configurator based the board selection.

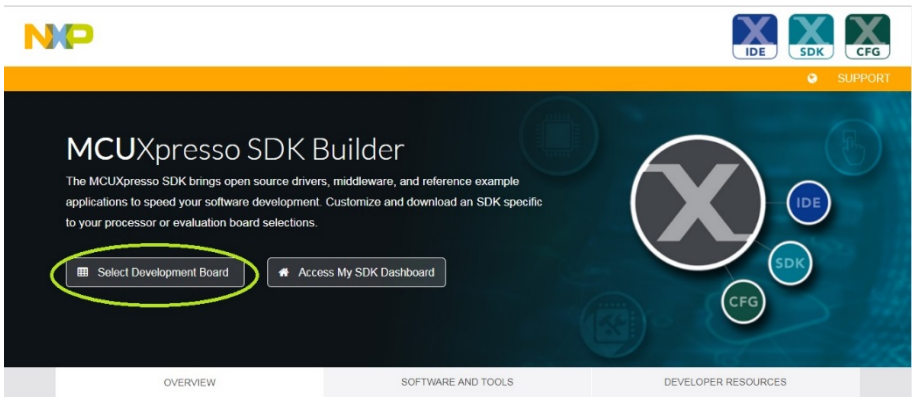

Fig 7. MCUXpresso SDK builder – board selection

3.) Sign in to the web page – it is necessary step

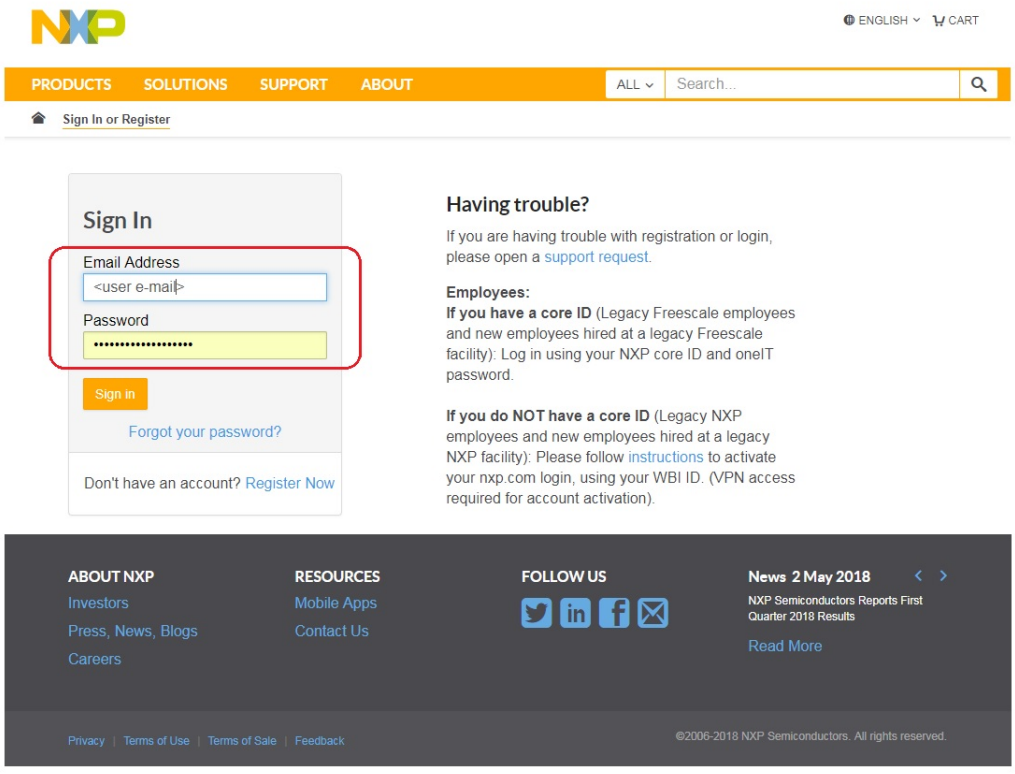

Fig 8. MCUXpresso login page

4.) Enter the keyword “FRDM-KW41Z“ and configure the SDK

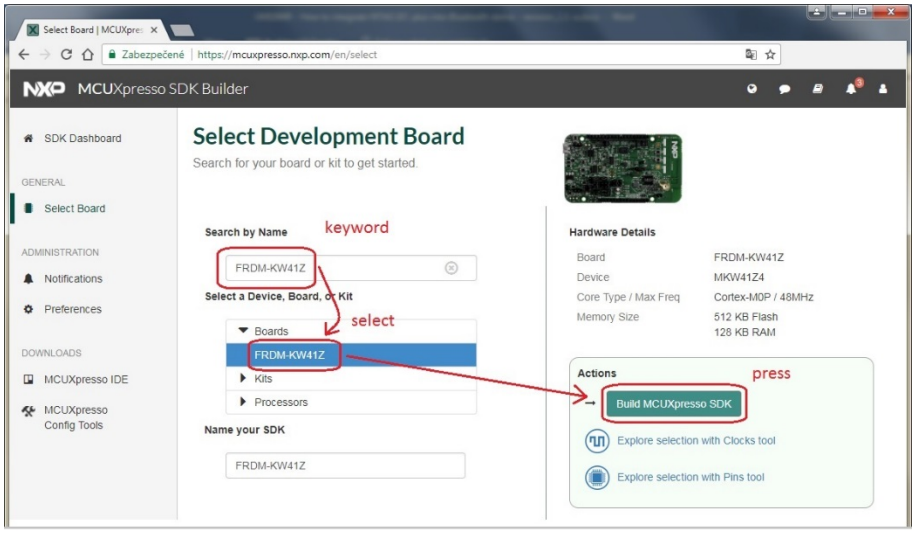

Fig 9. KW41Z board selection

5.) Select the SDK components

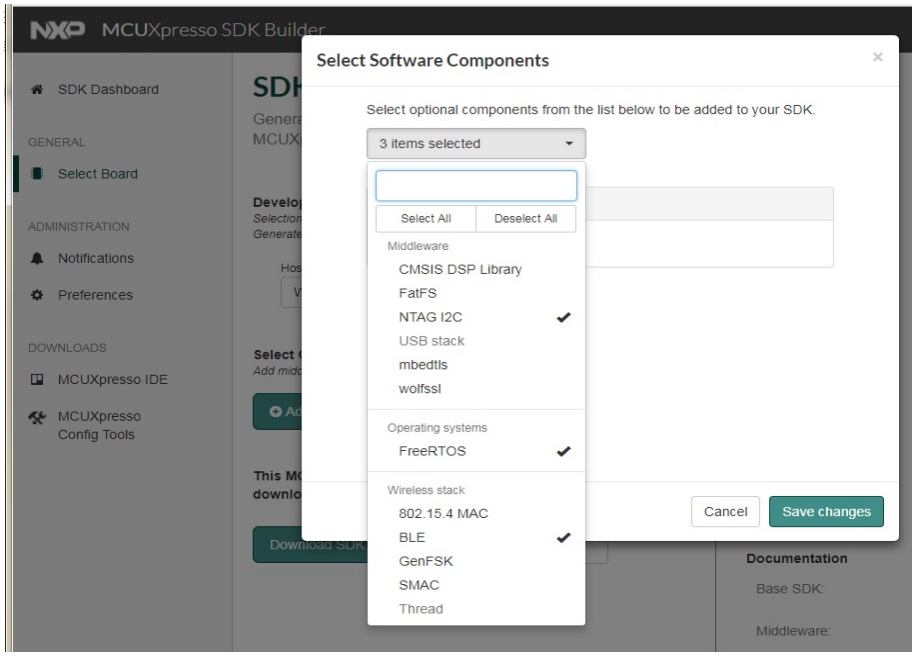

Fig 10. SDK components selection

6.) Verify and start SDK building

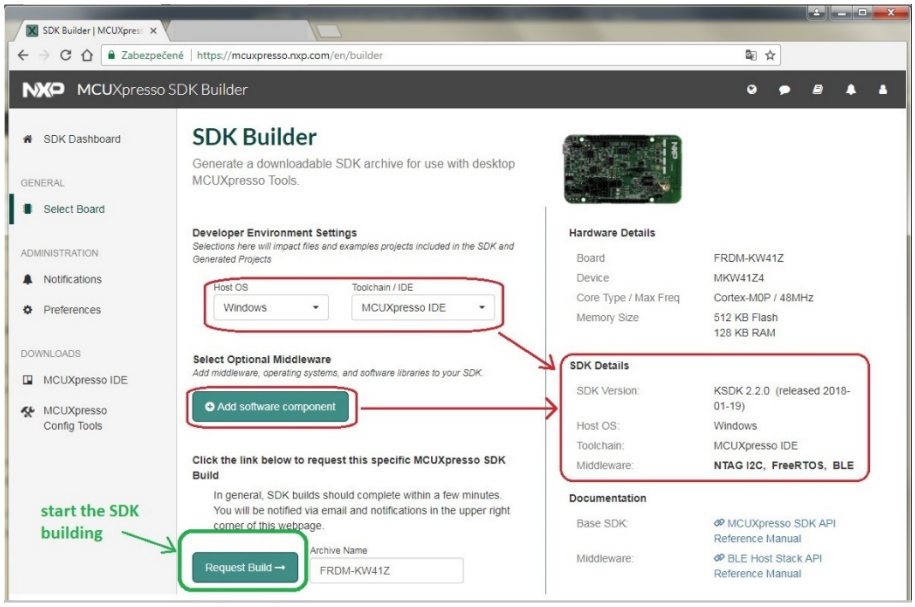

Fig 11. SDK creating

7.) Download the SDK

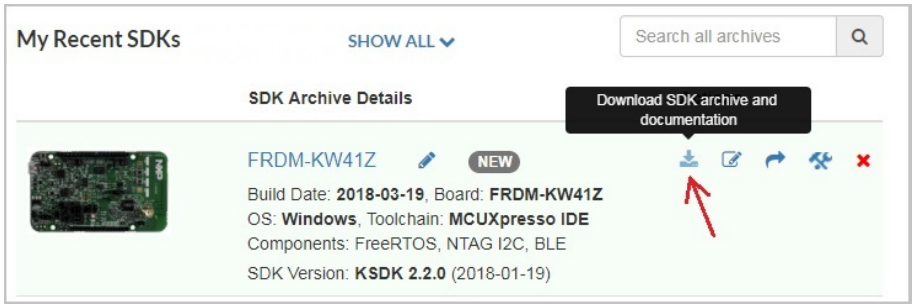

Fig 12. SDK downloading

2.1.4 Importing the SDK to the MCUXpresso IDE

The final step that needs to be implemented to achieve a fully functional development environment is importing the SDK. This step is very simple and it is done in a drag and drop way. The following procedure shows how to import SDK.

1.) Launch the MCUXpresso IDE and select the workspace via the occurred popup window.
2.) Switch window to the “Installed SDKs” and via drag & drop technique install the downloaded SDK (see figure Fig 13).

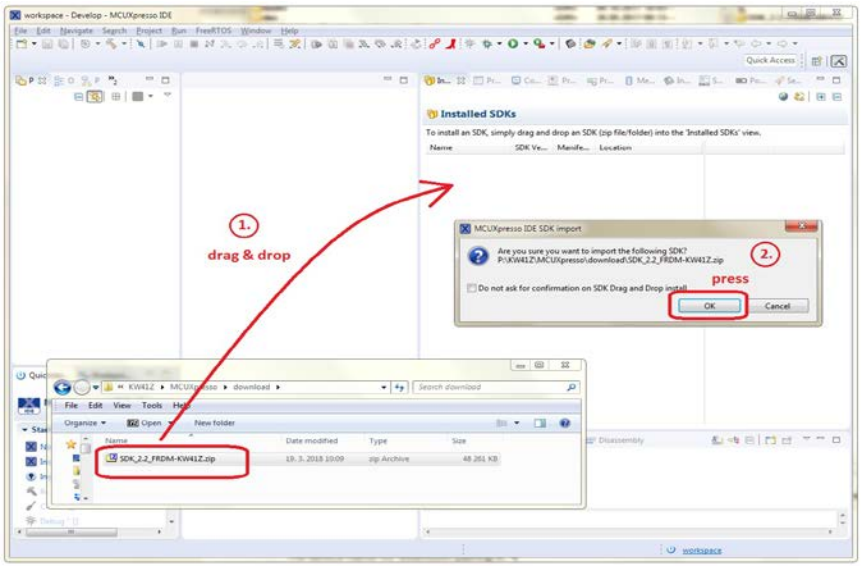

Fig 13. SDK installing to the MCUXpresso

2.2 How to show the demo

2.2.1 HID Demo from website

A ready-made version of the HID_device demo application is available on NXP website. The application consists of integrated *ntag_i2c_plus* middleware and already supports the BT pairing.

This sample project can be directly imported into MCUXpresso as an archived project. In the following chapter is the procedure how to debug this prepared project.

2.2.1.1 MCUXpresso and FRDM-KW41Z SDK

It is necessary to install the MCUXpresso, generate the FRDM-KW41Z SDK and import the SDK to MCUXpresso in accordance the chapter 2.1.

2.2.1.2 Importing of Archived Project

The archived project is packaged in the ZIP format. This project is then imported through the "Quickstart menu" and then "Import project(s) from file system..." (see Fig 14).

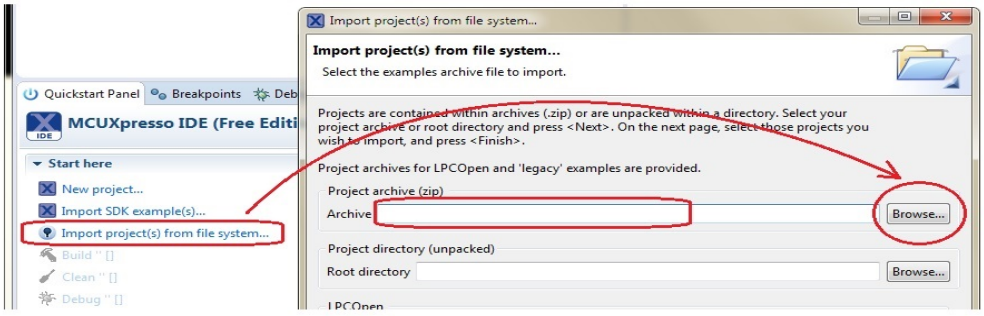

Fig 14. Setting the archived project

Find the archived project and continue importing. Project is automatically copied to the workspace.

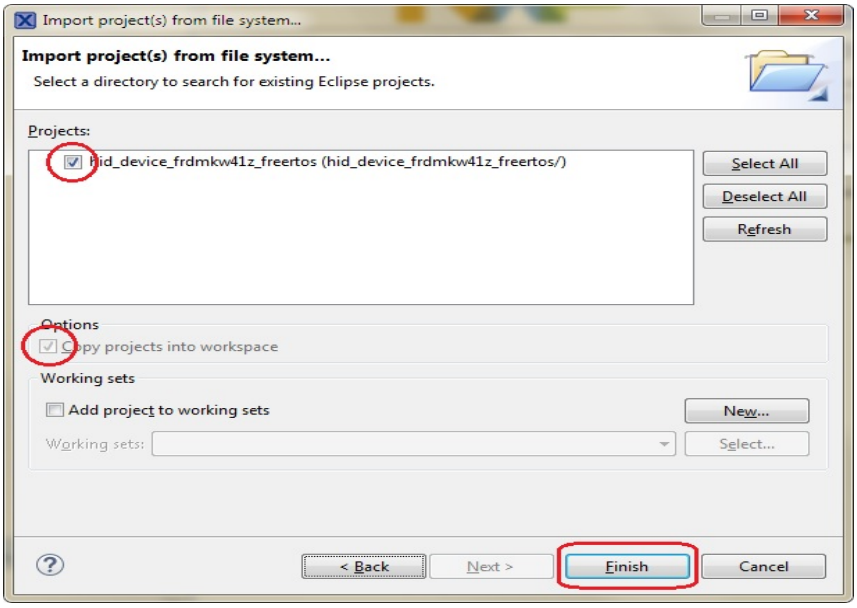

Fig 15. Importing of archived project

2.2.2 Demo running/debugging

Debug the application by clicking on the “Quickstart Panel” from the MCUXpresso on “Debug ’hid_device_frdmkw41z_freertos”. Accept the popup window of the J-link term of use.

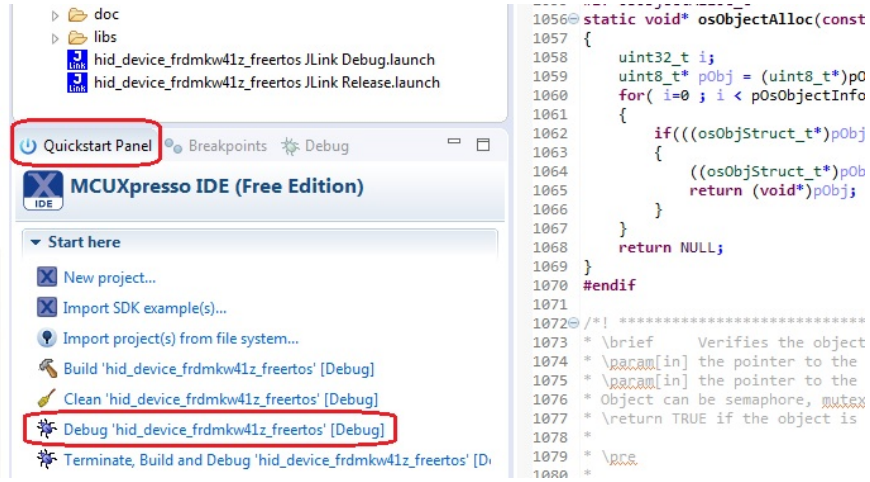

Fig 16. Demo compiling-flashing-debugging

The debugging starts via pressing the F8 on the keyboard (main menu tabs “Run->Resume”), see the Fig 17.

On the FRDM-KW41Z board, the white LED shall be blinking, notifying that the application is running and is in the standby mode.

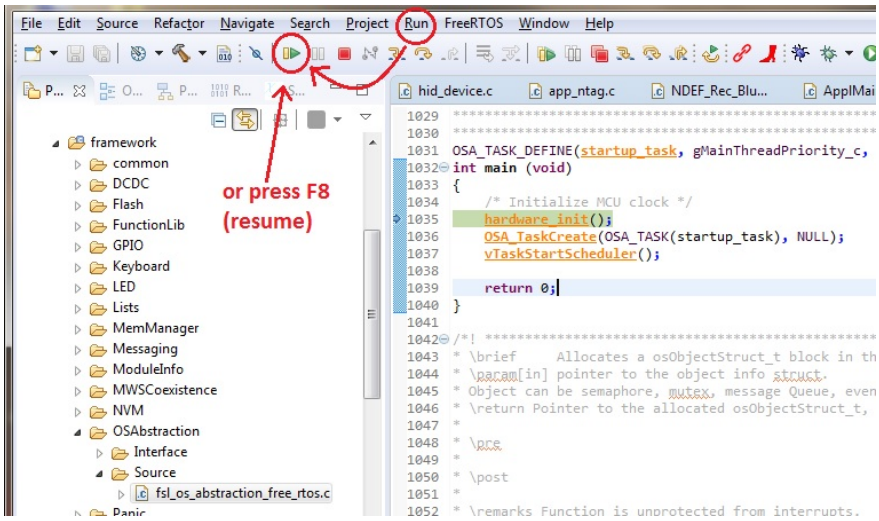

Fig 17. Start debugging

2.2.3 Behavior of the demo

By pressing the SW4 button for the first time after startup (now the red LED on KW41Z board is blinking). After all this the application is ready for usage or testing.

The following states describe the behavior of the HID_device demo application:

- *SW3 button pushed and phone is presented to the board:*
  The FRDM-KW41Z is paired over NFC automatically to the mobile phone. Depending on the version of Android, the application of FRDM-KW41Z is automatically connected and this is demonstrated by a moving cursor on your phone. For some versions of Android, it is needed to go into settings menu, BT-connections and tab on the paired BT-device. Then the connection to the application is finally performed.
  NOTE: This step was typically implemented from the mobile manufacturer as they wanted the user to define what the application is allowed to do.
- *SW4 button pushed and phone is presented to the board:*
  The NTAG I²C *plus* chip has now the URL of the NTAG I²C demo app written in an NDEF container. It automatically opens up either playstore or the app on the phone.

3. How to add NTAG I²C to the HID_device demo project

For using the NTAG I²C chip the *ntag_i2c_plus* middleware package should be added to the HID Bluetooth demo application. In the next chapters there is procedure how to add the *ntag_i2c_plus* middleware to the FRDM-KW41Z demo application.

3.1.1 HID_device – The demo application

The demo application, which we took as a basis for adding NTAG I²C is **hid_device**. This demo is based on operation system FreeRTOS or like “bare metal” (without operation system). We will take the version with the FreeRTOS. The demo shows a moving arrow on the smart phone display.

3.1.2 Import the HID_device demo application

The HID_device demo application is imported from the SDK. This step assumes that the MCUXpresso and SDK are installed in accordance with chapter 2.1. Then import the project will be done in the following steps.

1.) In the “Quickstart menu” click the “Import SDK example(s)…”. The “SDK Import Wizard” window is open. Select the required SKD (frdmkw41z) and press “Next”.

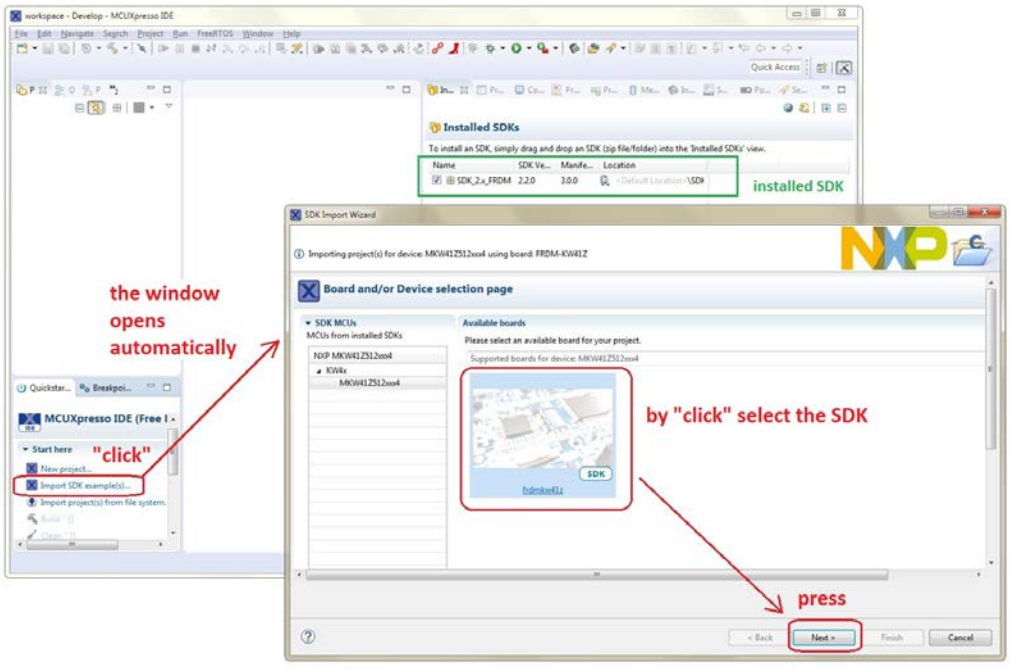

Fig 18. SDK selection

2.) In the “Examples” window search the “wireless_examples” and then “bluetooth” directory (see Fig 19).

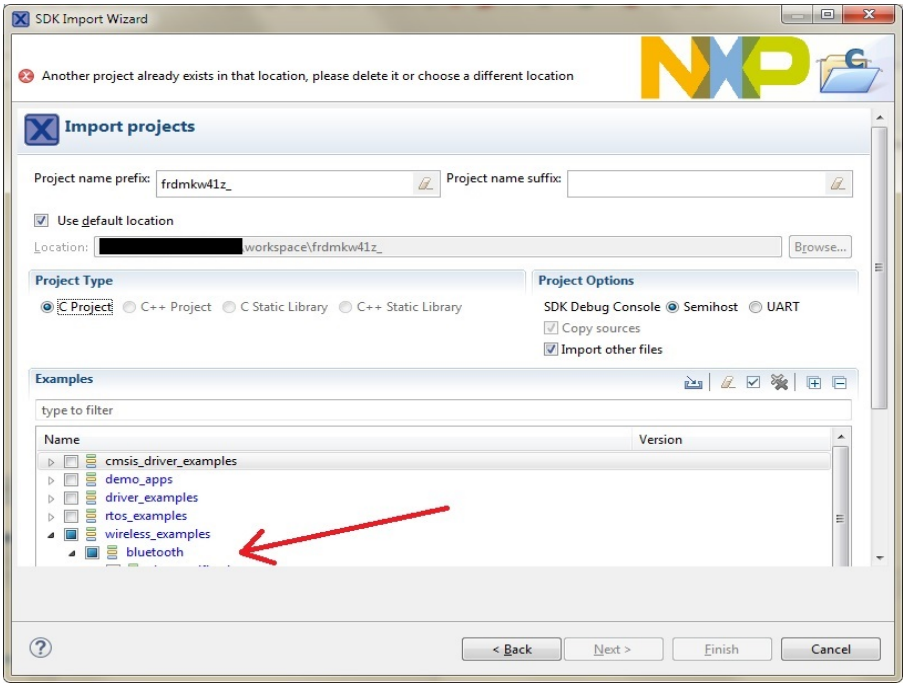

Fig 19. Finding the required project

3.) Then search “hid_device” demo application and select the freeRTOS version only.

4.) Press the “Finish” button and the project should be imported.

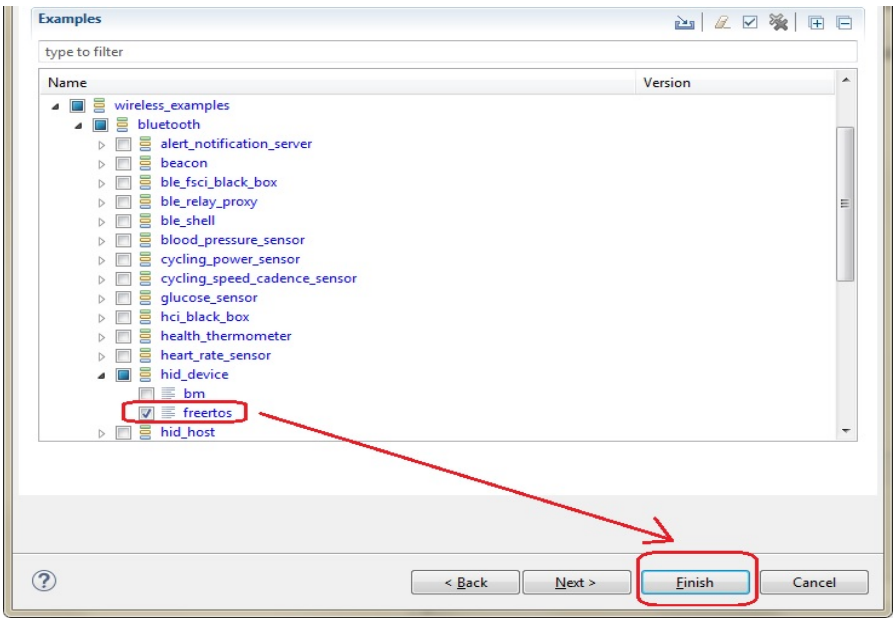

Fig 20. Importing the required project

3.1.3 "ntag_i2c_plus" middleware

The procedure how to add the NTAG middleware package is universal for all demo applications which supports FRDM-KW41Z development board. The name of the middleware package for NTAG I²C chip is **ntag_i2c_plus** and the actual version has the extension **ntag_i2c_plus_1.0.0**. It contains whole support software for NTAG I²C chip. The directory with middleware should be located at following directory path, directly in the SDK zip file:

SDK_2.2_FRDM-KW41Z.zip\middleware\ntag_i2c_plus_1.0.0

The internal structure of middleware should has following structure:

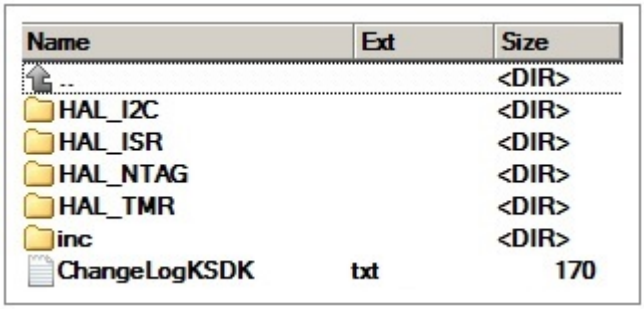

Fig 21. File structure of the ntag_i2c_plus_1.0.0

3.1.4 How to add the ntag_i2c_plus middleware to the Bluetooth demo

To have a HW support of the NTAG middleware SW there is necessary to add following to the Bluetooth demo application:

- setting for GPIO pins for communication interface I2C
- add the NTAG software handler declaration in to the application source C file
- implement the NTAG timer
- add #includes to the C sources of the BT demo application

NOTE: Parts of C code which have been added for support the NTAG I2C chip are separated by following conditional define:

```
#ifdef NTAG_I2C

#endif //NTAG_I2C
```

or following comment is added behind the C code, at the end of the line

```
#include "fsl_common.h" // added for NTAG middleware
```

3.1.4.1 GPIO pins setting

The NTAG I2C chip has, in addition to I2C pins, field detection (FD pin) and voltage out (Vout pin - energy harvesting). Only I2C bus is used to connect the NTAG I2C chip to the FRDM-KW41Z development kit. Other pins will remain unoccupied.

I²C pins

GPIO pins of the I²C interface are generally defined in the *pin_mux.c* file in the function *BOARD_InitI2C(void)* and should not be redefined in another location. GPIO pins for both I²C interfaces (I2C0 or I2C1) are set to I²C mode within this function.

3.1.4.2 NTAG SW and HW initialization

SW and HW initialization is called in the *main_task()* function in the *ApplMain.c* file.

First the HW initialization is performed by the function *hardware_init()* and there should be added following C-code, bellow the initialization DCDC module:

```
    /* Init DCDC module */
    BOARD_DCDCInit();

#ifdef NTAG_I2C
    /* Init I2C pins for NTAG communication */
    BOARD_InitI2C();
#endif // NTAG_I2C
```

Second is the SW initialization performed by function *HAL_I2C_InitDevice()* and *NTAG* I²C *handler (ntag_handle) is filled by the NFC_InitDevice() function.*

At the following C-code lines the SW initialization have to put before application thread calling (*App_Thread()*) in `main_task()` function.

```
#ifdef NTAG_I2C
  /* Initialize I2C for NTAG communication */
  HAL_I2C_InitDevice(HAL_I2C_INIT_DEFAULT, I2C_MASTER_CLK_SRC,
      NTAG_I2C_MASTER_BASEADDR);
  SystemCoreClockUpdate();

  /* Initialize the NTAG I2C components */
  ntag_handle = NFC_InitDevice((NTAG_ID_T)0, NTAG_I2C_MASTER_BASEADDR);
#endif // NTAG_I2C
  }

  /* Call application task */
  App_Thread( param );
}
```

The last step will be to insert the *ntag_handle* declaration at the beginning of the *ApplMain.c* file.

```
/*****************************************************************
* Public memory declarations
*****************************************************************/
#ifdef NTAG_I2C
NFC_HANDLE_T ntag_handle;          // NTAG handle
#endif // NTAG_I2C
```

3.1.4.3 Added #includes to the BLE demo application

Using the **"ntag_i2c_plus"** middleware requires include of headers into BLE demo application source code. Here is the list of files which require to include new headers:

ApplMain.c

```
#ifdef NTAG_I2C
/* NTAG middleware module */
#include "HAL_I2C_driver.h"
#include "app_ntag.h"
#endif //NTAG_I2C
```

hid_device.c

```
#ifdef NTAG_I2C
/* NTAG handler */
#include "app_ntag.h"
#endif //NTAG_I2C
```

3.1.4.4 Added new application NTAG files

There were created 2 application files (`app_ntag.c` and `app_ntag.h`) that creates an interface between NTAG middleware SW and common demo application.

The `app_ntag.c` source file contains sample functions for working with NDEF messages. Function `NFC_MsgWrite()` creates and writes the NDEF message in the Type-2 Tag format to the NTAG I²C chip through the ntag_i2c_plus middleware. The write algorithm is NFC-Forum compliance. Function `NDEF_Pairing_Write()` contains a procedure to create a BTSSP record via using the NDEF library. The same is performing function `NDEF_Demo_Write()` function. Here is shown how to create NDEF multi-record that contains several types of NDEF records.

The `app_ntag.h` header file contains predefined blocks of constants (constant fields of data) that are written to the NTAG I²C chip by default during the communication which requires set the default content to the chip's registers or erase the NTAG I²C chip user memory and registers of lock bytes.

3.1.5 BLE Demo Application Extension

The body and state machine of the *HID_device* demo application is in the `hid_device.c` file. Its extension for BLE pairing and writing NDEF messages to NTAG I²C chip is added to the function `BleApp_HandleKeys()`. This is because the writing of NDEF messages is done by pressing microswitch SW3 and SW4.

3.1.5.1 NDEF Timer

Within the extension of the HID_device demo application there was necessary to create NDEF timer. This one performs the time counter from the moment when SW3 or SW4 button is pressed. The timer counter is set to 2 seconds. During this time, it is indicated to write the NDEF administration to the NTAG I²C chip. The timer has no other function.

To add the timer, it is necessary to add the following C-code to the `hid_device.c` file:

Add the declaration of the timer handler

This declaration is placed at the beginning to the `hid_device.c` file in to the part "Private memory declarations".

```
/************************************************************************************
* Private memory declarations
```

```
*****************************************************************************/
#ifdef NTAG_I2C
static tmrTimerID_t mNDEFTimerId;
static bool boNDEFState = FALSE;
#endif
```

Add the declaration of the timer callback function

NDEF timer callback function declaration is placed to the part “Private functions prototypes”.

```
/*****************************************************************************
* Private functions prototypes
*****************************************************************************/
#ifdef NTAG_I2C
static void NDEFTimerCallback(void *);
#endif
```

Allocate / Initialize the timer

There are 3 timers used within the HID_device demo application. The NDEF timer is also necessary to allocate in the function `BleApp_Config()` in the `hid_device.c` file, at the same place as the common timers are allocated. Function `TMR_AllocateTimer()` returns timer ID value which is stored in the variable `mNDEFTimerId`. The timer ID allocation must be added behind the other timer as it is done at following C-code printout

```
    /* Allocate application timers */
    mAdvTimerId = TMR_AllocateTimer();
    mHidDemoTimerId = TMR_AllocateTimer();
    mBatteryMeasurementTimerId = TMR_AllocateTimer();

#ifdef NTAG_I2C
    mNDEFTimerId = TMR_AllocateTimer();
#endif
```

Add the timer callback function

There is necessary add the *`NDEFTimerCallback()`* function at the end of the *`hid_device.c`* file. If NDEF timer counter expires timer is stop. Then RGB LED is switched off. There is the printout of the call back function at the following lines.

```
#ifdef NTAG_I2C
/*! *****************************************************************************
 * \brief      Handles timer callback for writing NDEF messages
 *
 * \param[in]   pParam      callback parameters.
*
 ***************************************************************************** */
static void NDEFTimerCallback(void * pParam)
  {
    /* Stop Advertising Timer */
    TMR_StopTimer(mNDEFTimerId);
    /* switch off the LED indication */
    TurnOffLeds ();
  }
#endif // NTAG_I2C
```

3.1.6 Security change

The sample project for adding NTAG I²C middleware is **hid_device** and is described in chapter 3.1.1. This project requires to enter the password “999999” during the Bluetooth pairing. From this reason is necessary to degrease the security level to remove the password sequence.

Security level is a part of the configuration and is set in the *app_config.c* file. In this file following parameter must be changed

gSecurityMode_1_Level_3_c

to the new parameter

gSecurityMode_1_Level_1_c

Parameter *gSecurityMode_1_Level_3_c* is used on several places within the *app_config.c* file. Use the *FIND* function (short key is “CTRL+F”) of the KDS IDE to find it and update.

There are last two parameters of the `gPairingParameters` structure which are necessary to change.

parameter:

.securityModeAndLevel = gSecurityMode_1_Level_3_c,

has to be changed to:

.securityModeAndLevel = gSecurityMode_1_Level_1_c,

parameter:

.localIoCapabilities = gIoDisplayOnly_c,

has to be changed to:

.localIoCapabilities = gIoNone_c,

parameter

.leSecureConnectionSupported = TRUE,

has to be changed to

.leSecureConnectionSupported = FALSE,

3.2 HID_device project properties

3.2.1 Symbols

Within the project setting is necessary to add following “symbols”:

- FRDM_KW41Z
- NTAG_I2C
- I2C_FSL
- HAVE_STDBOOL_H

These symbols are conditional defines for compiler and allows using of the *ntag_i2c_plus* middleware and allows to add required GPIO pins configuration for HW connection with NTAG I²C *plus* PCB board. Following picture shows the place where the symbols are located within the project properties.

There are changes for NDEF library by the green color. This is described in the next chapter 0.

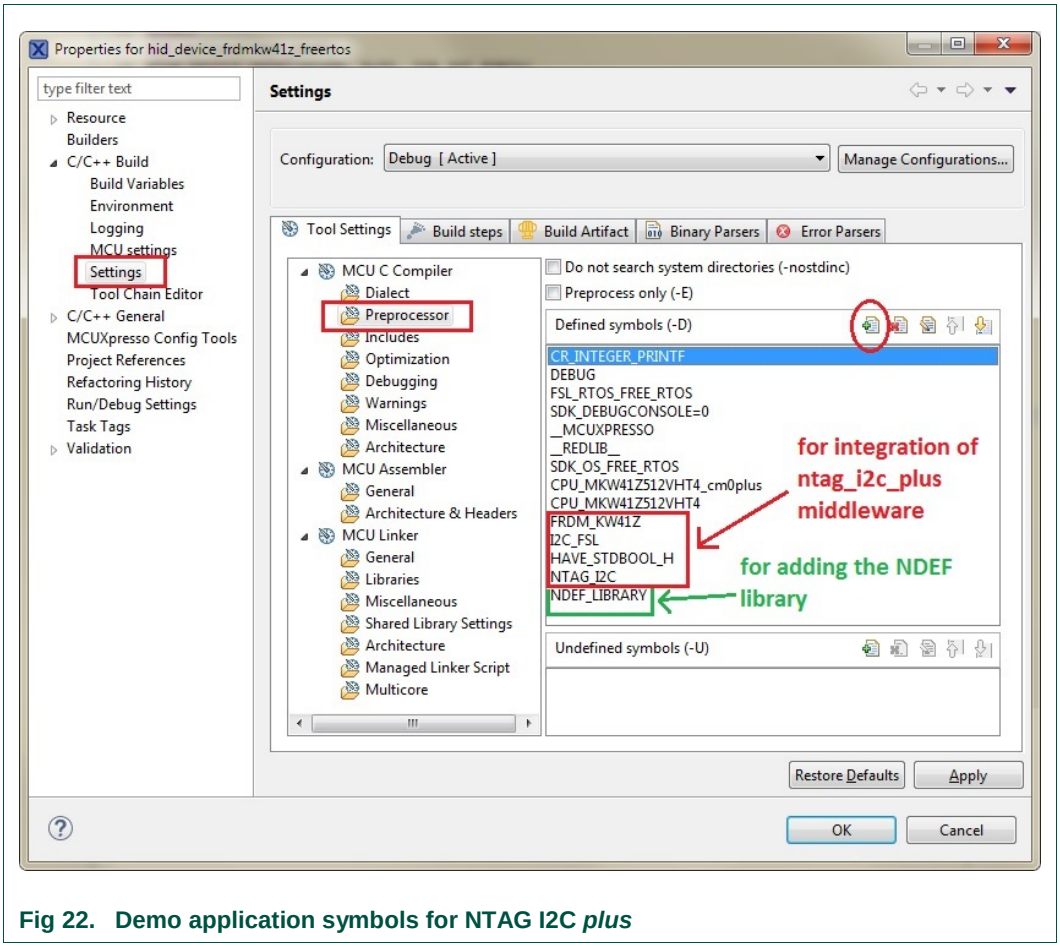

Fig 22. Demo application symbols for NTAG I2C *plus*

3.2.2 Include paths

Within the project setting is necessary to add following “includes”:

- "${workspace_loc:/${ProjName}/ntag_i2c_plus_1.0.0/inc}"
- "${workspace_loc:/${ProjName}/ntag_i2c_plus_1.0.0/HAL_I2C/inc}"
- "${workspace_loc:/${ProjName}/ntag_i2c_plus_1.0.0/HAL_NTAG}"
- "${workspace_loc:/${ProjName}/ntag_i2c_plus_1.0.0/HAL_NTAG/inc}"
- "${workspace_loc:/${ProjName}/ntag_i2c_plus_1.0.0/HAL_ISR}"
- "${workspace_loc:/${ProjName}/ntag_i2c_plus_1.0.0/HAL_ISR/inc}"
- "${workspace_loc:/${ProjName}/ntag_i2c_plus_1.0.0/HAL_TMR/inc}"

These includes represent the paths which point to source files of the *ntag_i2c_plus* middleware. Following picture shows the place where the includes are located within the project properties.

There are changes for NDEF library by the green color. This is described in the next chapter 0.

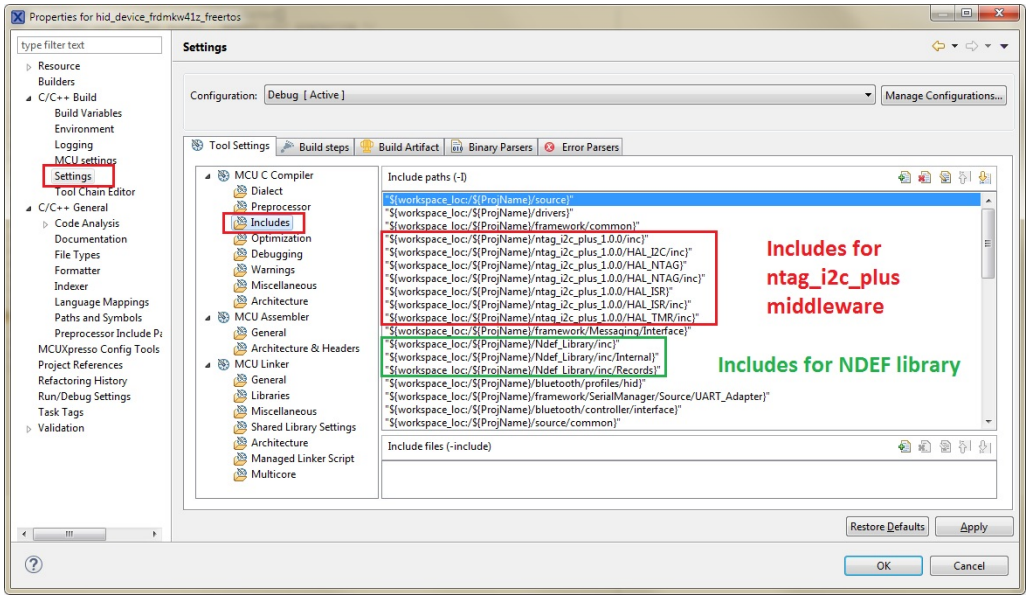

Fig 23. Demo application includes for NTAG I2C *plus*

3.3 NDEF library usage

The NDEF library supports the use of NDEF messages and greatly facilitates it. This library is used to create NDEF records and to encode and decode NDEF messages. The NDEF library is not currently available in the generated SDK from the NXP website. Its basic version is part of only this BLE pairing HID_device demo application package.

The library is added to the project as uncompiled. This implies the need to add a not a library binary file, but the entire directory with its source code to the HID_device demo application. From this reason is necessary to set the Symbols and include the paths to the MCUXpresso project of the HID_device demo application.

3.3.1 How to add the NDEF library to the Bluetooth demo

For this demo, the NDEF library functions are used only within the *app_ntag.c* file. Therefore, it is necessary to add a following includes to the file.

```
/* NDEF library headers */
#ifdef NDEF_LIBRARY
#include "NDEF_Rec_Text.h"
#include "NDEF_Rec_Uri.h"
#include "NDEF_Rec_BluetoothSsp.h"
#include "NDEF_Rec_Aar.h"
#include "NDEF_Message.h"
#include "NDEF_Rec_GenericPayload.h"
#include "NDEF_Record.h"
#endif // NDEF_LIBRARY
```

3.3.2 Symbols

Within the project setting is necessary to add only one “symbols”.

- NDEF_LIBRARY

The setting where to place it is in the picture in chapter 3.2.1

3.3.3 Include paths

Within the project setting is necessary to add following “includes”:

- "${workspace_loc:/${ProjName}/Ndef_Library/inc}"
- "${workspace_loc:/${ProjName}/Ndef_Library/inc/Internal}"
- "${workspace_loc:/${ProjName}/Ndef_Library/inc/Records}"

The setting where to place it is in the picture in chapter 3.2.2

3.3.4 How not to use the NDEF library

An option for not using the NDEF library is added to the HID_device demo application. NDEF messages (the BTSSP and smart poster about the NTAG I²C plus demo board) are hardcoded in this case and the C-code is already written in the *app_ntag.c* file (see chapter 0). This can be done if **NDEF_LIBRARY** define is not defined.

Then the hardcoded NDEF message is processed in to the standard Tag-2 Type format and written to the NTAG I²C chip. There is no difference to the case when the NDEF library is used.

NOTE 1: The BTSSP NDEF message is also described by the comments. The description consists only of the mandatory parameter and necessary optional data, which are required for BT pairing. This is mainly from reason to very quickly understand the BTSSP NDEF message format.

4. Abbreviations

Table 2. Abbreviations

Acronym	Description
BLE	Bluetooth Low Energy – standard 4.2.
BTSSP	Bluetooth Secure Simple Pairing (NDEF record or message)
CMSIS	Cortex Microcontroller Software Interface Standard
FLASH	an electronic non-volatile computer storage medium
FRDM-KW41	Freedom board development kit based on MKW41Z microcontroller
GPIO	General Purpose Input Output
HW	Hardware
IDE	Integrated Development Environment
IRQ	Interrupt Request
MAC address	Media Access Control address
MCU	Microcontroller Unit
NDEF	NFC Data Exchange Format
NFC	Near Field Communication
OS	Operation System
PCB	Printed Circuit Board
RGB LED	Full color LED (Red-Green-Blue)
SDK	Software Development Kit
SW	Software

5. References

[1] **FRDM-KW41Z: NXP® Freedom Development Kit for Kinetis® KW41Z/31Z/21Z MCUs:**
http://www.nxp.com/products/software-and-tools/hardware-development-tools/freedom-development-boards/nxp-freedom-development-kit-for-kinetis-kw41z-31z-21z-mcus:FRDM-KW41Z?fsrch=1&sr=1&pageNum=1

[2] **NTAG I²C *plus* Explorer Kit (OM5569) general NXP web site:**
http://www.nxp.com/products/identification-and-security/nfc-and-reader-ics/connected-tag-solutions/ntag-ic-plus-explorer-kit-demo-kit:OM5569-NT322E

[3] **OM23221ARD: NTAG I²C *plus* kit for Arduino® pinout**
https://www.nxp.com/products/identification-and-security/nfc/nfc-tags-for-electronics/ntag-ic-iplus-i-kit-for-arduino-pinout:OM23221ARD?fsrch=1&sr=1&pageNum=1

[4] **MCUXpresso SDK Builder**
https://mcuxpresso.nxp.com/en/welcome

.

6. Legal information

6.1 Definitions

Draft — The document is a draft version only. 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 herein and shall have no liability for the consequences of use of such information.

6.2 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.

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.

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

Evaluation products — This product is provided on an "as is" and "with all faults" basis for evaluation purposes only. NXP Semiconductors, its affiliates and their suppliers expressly disclaim all warranties, whether express, implied or statutory, including but not limited to the implied warranties of non-infringement, merchantability and fitness for a particular purpose. The entire risk as to the quality, or arising out of the use or performance, of this product remains with customer.

In no event shall NXP Semiconductors, its affiliates or their suppliers be liable to customer for any special, indirect, consequential, punitive or incidental damages (including without limitation damages for loss of business, business interruption, loss of use, loss of data or information, and the like) arising out the use of or inability to use the product, whether or not based on tort (including negligence), strict liability, breach of contract, breach of warranty or any other theory, even if advised of the possibility of such damages.

Notwithstanding any damages that customer might incur for any reason whatsoever (including without limitation, all damages referenced above and all direct or general damages), the entire liability of NXP Semiconductors, its affiliates and their suppliers and customer's exclusive remedy for all of the foregoing shall be limited to actual damages incurred by customer based on reasonable reliance up to the greater of the amount actually paid by customer for the product or five dollars (US$5.00). The foregoing limitations, exclusions and disclaimers shall apply to the maximum extent permitted by applicable law, even if any remedy fails of its essential purpose.

6.3 Licenses

Purchase of NXP ICs with NFC technology

Purchase of an NXP Semiconductors IC that complies with one of the Near Field Communication (NFC) standards ISO/IEC 18092 and ISO/IEC 21481 does not convey an implied license under any patent right infringed by implementation of any of those standards. Purchase of NXP Semiconductors IC does not include a license to any NXP patent (or other IP right) covering combinations of those products with other products, whether hardware or software.

6.4 Trademarks

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

NTAG — is a trademark of NXP B.V.

I²C-bus — is a trademark of NXP B.V.

7. List of figures

8. Contents

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

Date of release: 5 June 2018 **422220**

Document identifier: UM10950