UM10949

How to integrate NTAG I²C *plus* into Kinetis KW41 - Bluetooth demo

Rev. 1.0 — 7 March 2017 User manual

422110 COMPANY PUBLIC

Document information

Info	Content
Keywords	NTAG I²C *plus*, Kinetis KW41, Bluetooth, Bluetooth BLE
Abstract	This document gives a description on how to integrate NTAG I²C *plus* middleware into the Kinetis microcontroller firmware.

Revision history

Rev	Date	Description
1.0	20170307	Initial version

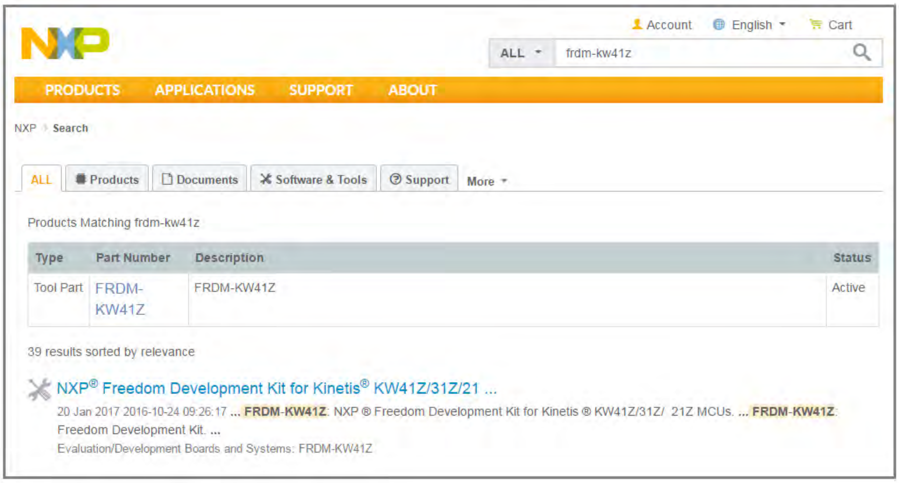
Contact information

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

1 FRDM-KW41Z Startup

All documentation and tutorials for the FRDM-KW41Z development board startup is available on the NXP website [1.]. For this reason, here is only info how to find this information. Here are required steps:

- Startup procedure is located on the www.nxp.com website.
- Search the expression "FRDM-KW41Z"
- Select the link shown on the following picture.



Products Matching frdm-kw41z

Type	Part Number	Description	Status
Tool Part	FRDM-KW41Z	FRDM-KW41Z	Active

39 results sorted by relevance

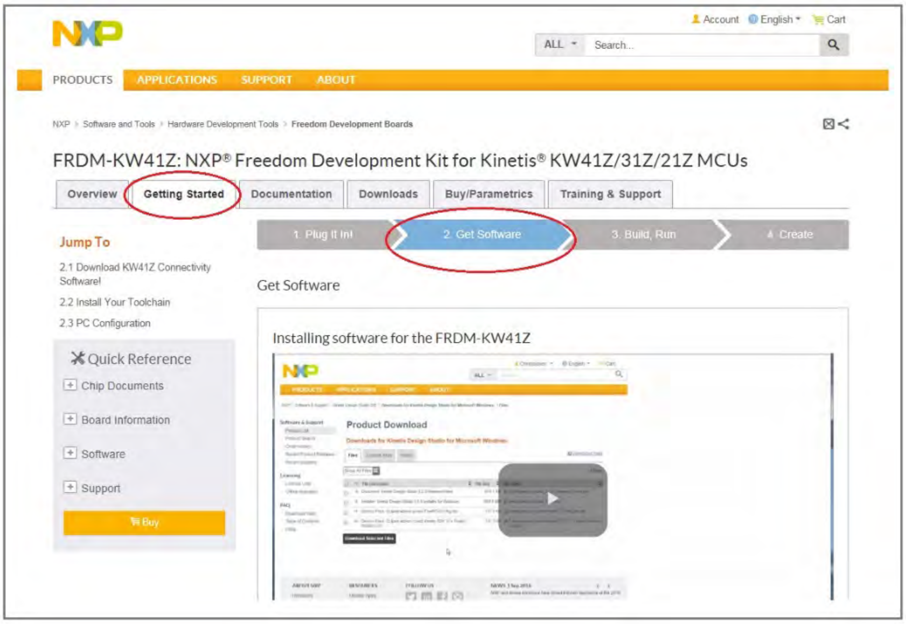
NXP® Freedom Development Kit for Kinetis® KW41Z/31Z/21 ...

20 Jan 2017 2016-10-24 09:26:17 ... FRDM-KW41Z: NXP ® Freedom Development Kit for Kinetis ® KW41Z/31Z/ 21Z MCUs. ... FRDM-KW41Z: Freedom Development Kit. ...

Evaluation/Development Boards and Systems: FRDM-KW41Z

1.1 Toolchain Installation

Toolchain installation (its actual version) is described within the second step of the "Getting Started" tab. This procedure is available on the NXP website (see following picture).



FRDM-KW41Z: NXP® Freedom Development Kit for Kinetis® KW41Z/31Z/21Z MCUs

Jump To

2.1 Download KW41Z Connectivity Software!

2.2 Install Your Toolchain

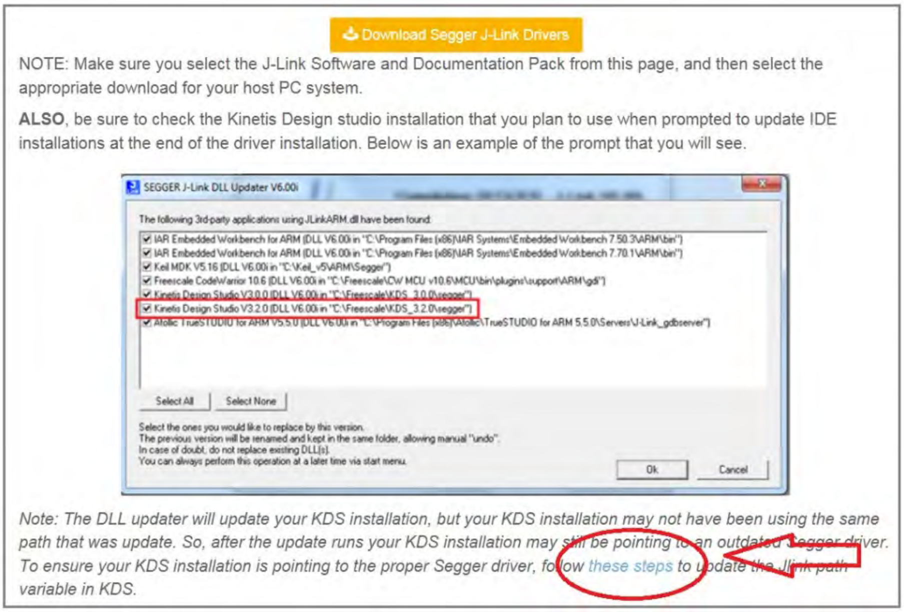
2.3 PC Configuration

Get Software

Installing software for the FRDM-KW41Z

1.1.1 Segger J-Link driver

After installation of the Segger J-Link driver it could happen that this driver is not available within the KDS. There is a written procedure how to fix this problem. You can find it by using the website link under the “these steps” marker at the screenshot below.



NOTE: Make sure you select the J-Link Software and Documentation Pack from this page, and then select the appropriate download for your host PC system.

ALSO, be sure to check the Kinetis Design studio installation that you plan to use when prompted to update IDE installations at the end of the driver installation. Below is an example of the prompt that you will see.

SEGGER J-Link DLL Updater V6.00i

The following 3rd-party applications using JLinkARM.dll have been found:

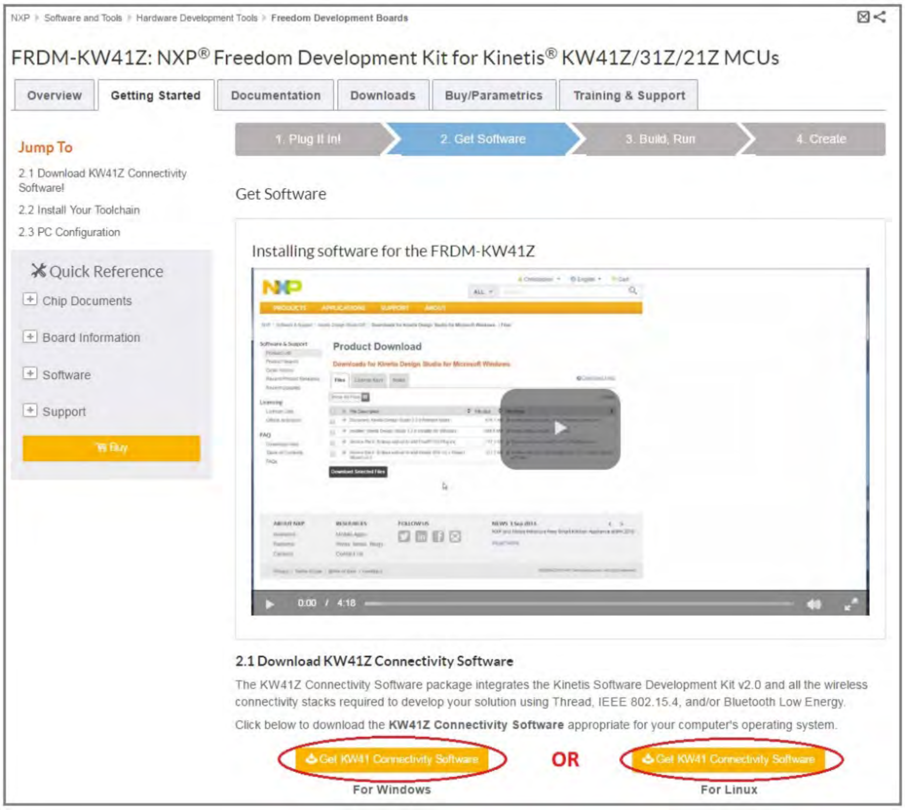
- ☑ IAR Embedded Workbench for ARM (DLL V6.00i in "C:\Program Files (x86)\IAR Systems\Embedded Workbench 7.50.3\ARM\bin")
- ☑ IAR Embedded Workbench for ARM (DLL V6.00i in "C:\Program Files (x86)\IAR Systems\Embedded Workbench 7.70.1\ARM\bin")
- ☑ Keil MDK V5.16 (DLL V6.00i in "C:\Keil_v5\ARM\Segger")
- ☑ Freescale CodeWarrior 10.6 (DLL V6.00i in "C:\Freescale\CW MCU v10.6\MCU\bin\plugins\support\ARM\gdi")
- ☑ Kinetis Design Studio V3.0.0 (DLL V6.00i in "C:\Freescale\KDS_3.0.0\segger")
- ☑ Kinetis Design Studio V3.2.0 (DLL V6.00i in "C:\Freescale\KDS_3.2.0\segger")
- ☑ Atollic TrueSTUDIO for ARM V5.5.0 (DLL V6.00i in "C:\Program Files (x86)\Atollic\TrueSTUDIO for ARM 5.5.0\Servers\J-Link_gdbserver")

Select the ones you would like to replace by this version.
The previous version will be renamed and kept in the same folder, allowing manual "undo".
In case of doubt, do not replace existing DLL(s).
You can always perform this operation at a later time via start menu.

Note: The DLL updater will update your KDS installation, but your KDS installation may not have been using the same path that was update. So, after the update runs your KDS installation may still be pointing to an outdated Segger driver. To ensure your KDS installation is pointing to the proper Segger driver, follow these steps to update the Jlink path variable in KDS.

1.1.2 How to get SDK - Kinetis KW41Z Connectivity Software

Kinetis KW41Z Connectivity Software is the software development kit directly for the KW41Z platform. This SDK is not a part of the general SDK base which covers SW base for many NXP microcontrollers. Kinetis KW41Z Connectivity Software is separated installation file which have to be downloaded from the NXP web site and installed on the desktop PC. The procedure is following.



FRDM-KW41Z: NXP® Freedom Development Kit for Kinetis® KW41Z/31Z/21Z MCUs

Get Software

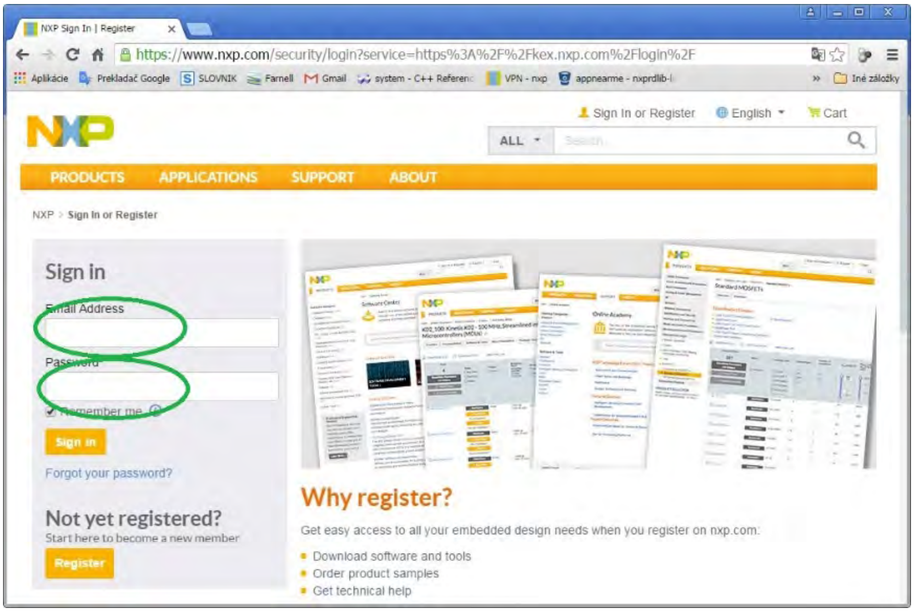
Installing software for the FRDM-KW41Z

2.1 Download KW41Z Connectivity Software

The KW41Z Connectivity Software package integrates the Kinetis Software Development Kit v2.0 and all the wireless connectivity stacks required to develop your solution using Thread, IEEE 802.15.4, and/or Bluetooth Low Energy.

Click below to download the **KW41Z Connectivity Software** appropriate for your computer's operating system.

Download the installation file for your OS. Downloading requires sign in process via user account.

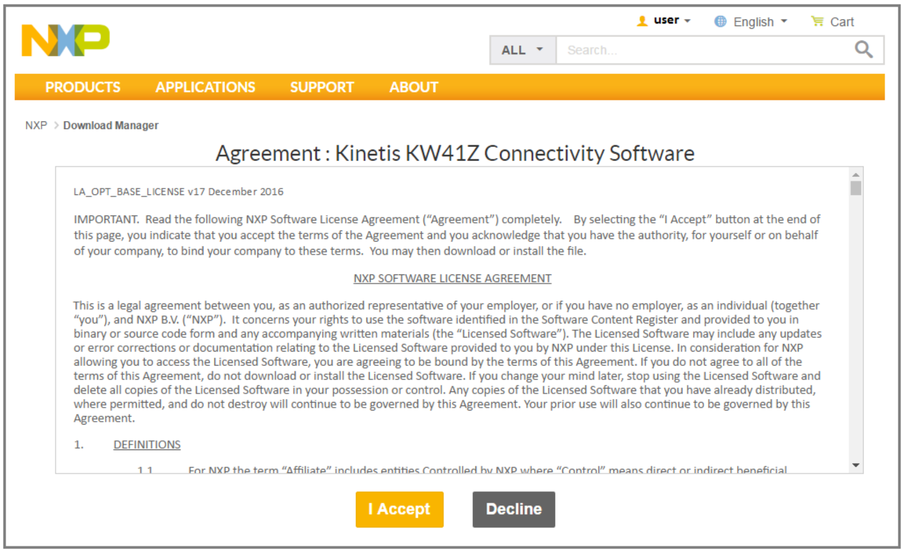


Not yet registered?
Start here to become a new member

Why register?

Get easy access to all your embedded design needs when you register on nxp.com:

- Download software and tools
- Order product samples
- Get technical help



Agreement : Kinetis KW41Z Connectivity Software

LA_OPT_BASE_LICENSE v17 December 2016

IMPORTANT. Read the following NXP Software License Agreement (“Agreement”) completely. By selecting the “I Accept” button at the end of this page, you indicate that you accept the terms of the Agreement and you acknowledge that you have the authority, for yourself or on behalf of your company, to bind your company to these terms. You may then download or install the file.

NXP SOFTWARE LICENSE AGREEMENT

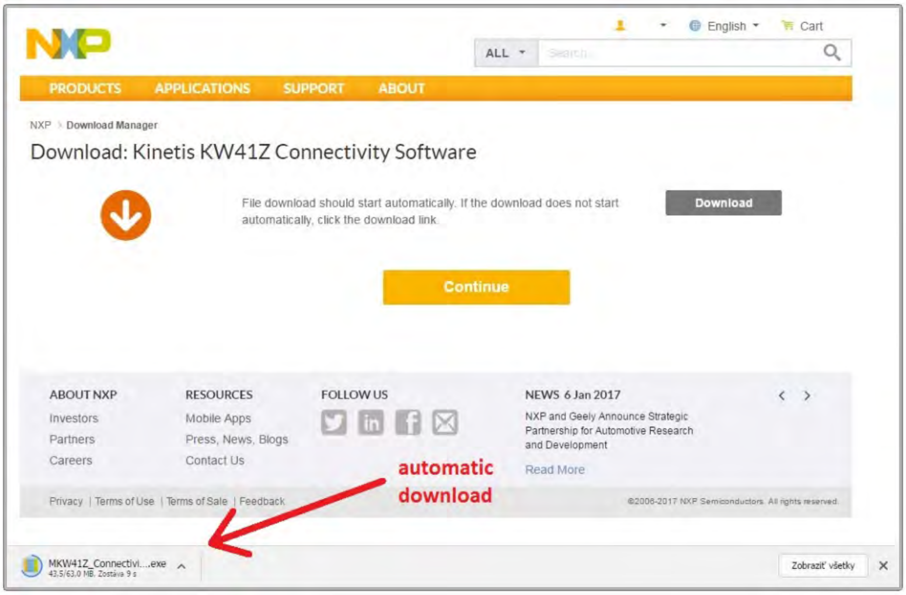
This is a legal agreement between you, as an authorized representative of your employer, or if you have no employer, as an individual (together “you”), and NXP B.V. (“NXP”). It concerns your rights to use the software identified in the Software Content Register and provided to you in binary or source code form and any accompanying written materials (the “Licensed Software”). The Licensed Software may include any updates or error corrections or documentation relating to the Licensed Software provided to you by NXP under this License. In consideration for NXP allowing you to access the Licensed Software, you are agreeing to be bound by the terms of this Agreement. If you do not agree to all of the terms of this Agreement, do not download or install the Licensed Software. If you change your mind later, stop using the Licensed Software and delete all copies of the Licensed Software in your possession or control. Any copies of the Licensed Software that you have already distributed, where permitted, and do not destroy will continue to be governed by this Agreement. Your prior use will also continue to be governed by this Agreement.

1. DEFINITIONS

1.1 For NXP the term “Affiliate” includes entities Controlled by NXP where “Control” means direct or indirect beneficial

I Accept Decline

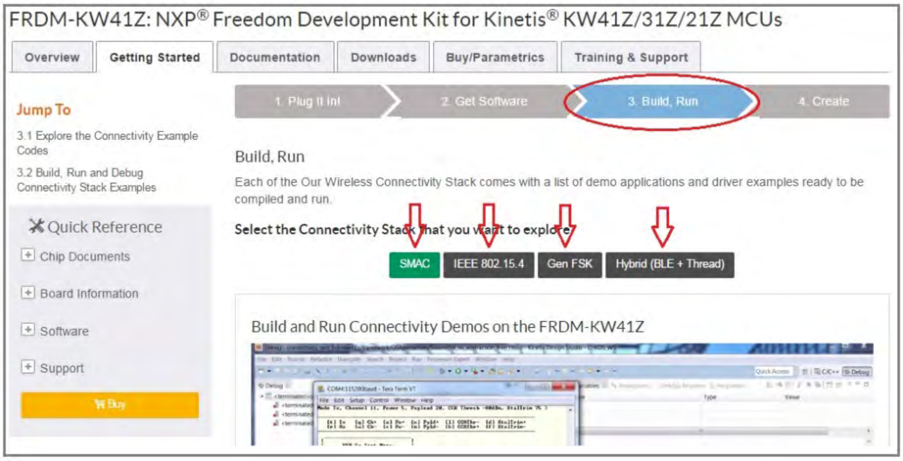
Accept the license agreement and the automatic download is started.



Download: Kinetis KW41Z Connectivity Software

File download should start automatically. If the download does not start automatically, click the download link.

When the file MKW41Z_Connectivity_Software.exe is downloaded, install the SDK to the recommended default place. After installation projects for FRDM-KW41 development board could be imported to the user workspace. Build and debug process of the demo applications are described within the third step of the “Getting Started” tab. It is possible to select the education video for different connectivity stacks. See the following picture.



FRDM-KW41Z: NXP® Freedom Development Kit for Kinetis® KW41Z/31Z/21Z MCUs

Build, Run

Each of the Our Wireless Connectivity Stack comes with a list of demo applications and driver examples ready to be compiled and run.

Select the Connectivity Stack that you want to explore

Build and Run Connectivity Demos on the FRDM-KW41Z

1.1.3 How to add NTAG I2C to the BLE project

For using the NTAG I²C chip the middleware software package should be added to the Bluetooth demo application. The NTAG middleware software package should be a part of whole middleware software package for the FRDM-KW41Z board in the near future, but by now it is available as package on Arduino page. Next chapters describe manual how to add the NTAG middleware to the FRDM-KW41Z demo application.

1.1.4 FSL_HID – The Sample Project

Sample project, which we took as a basis for adding NTAG I²C is **hid_device**. This project is located at the following directory path:

.\MKW41Z_ConnSw_1.0.2\boards\frdmkw41z\wireless_examples\bluetooth\hid_device

The device name for Bluetooth pairing is:

FSL_HID

The MAC address of the FRDM_KW41Z is:

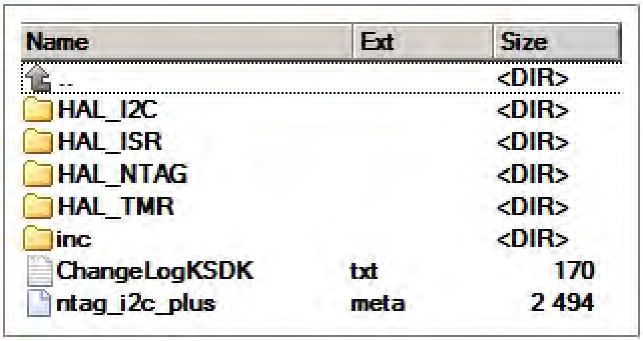
00:04:9F:00:00:04

1.1.5 ntag_i2c_plus middleware

The procedure how to add the NTAG middleware SW package is universal for all demo applications which supports FRDM-KW41Z and USB-KW41Z development boards. The name of the middleware SW package for NTAG I²C chip is **“ntag_i2c_plus”**. It contains whole support software for NTAG I²C chip. The directory with middleware should be located at following directory path:

.\MKW41Z_ConnSw_1.0.2\middleware\ntag_i2c_plus

In case that ntag_i2c_plus middleware software is missing at this path, please just copy it here from the delivered software package (SW4223_2017-01-30-SDK_FRDM-KW41Z.zip). Then the internal structure of middleware should have following structure:



Name	Ext	Size
..		<DIR>
HAL_I2C		<DIR>
HAL_ISR		<DIR>
HAL_NTAG		<DIR>
HAL_TMR		<DIR>
inc		<DIR>
ChangeLogKSDK	txt	170
ntag_i2c_plus	meta	2 494

1.1.6 How to Add the ntag_i2c_plus Middleware to the BLE Project

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
- setting for FD (field detection) GPIO pin
- 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 BLE 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
```

1.1.6.1 GPIO pins setting

I²C pins

GPIO pins of the I²C interface are generally defined in the *pin_mux.c* file and should not be redefined in another location. There is a function *configure_i2c_pins()* which sets the required number of I²C interface (I2C0 or I2C1) and sets the right mode of the pins for signals SCL and SDA.

FD pin

FD GPIO pin represents NFC field detection. This is output pin on the NTAG I²C and input pin for the FRDM-KW41Z. Function of this pin is not used in NTAG demo software [2.] and it also not used within adding to BLE. However the whole support for this pin exists in NTAG demo software and is added also to BLE demo application.

FD pin is input pin for MCU and there is necessary to add following initialization function to the *pin_mux.c* file:

```
#ifdef NTAG_I2C
// initialization FD pin
```

```
#define PIN17_IDX                    17u   /*!< Pin number for pin 17 in a port */

/*
 * TEXT BELOW IS USED AS SETTING FOR THE PINS TOOL
****************************
BOARD_InitPins:
- options: {coreID: singlecore, enableClock: 'true'}
- pin_list:
  - { peripheral: GPIOC, signal: 'GPIO, 17', pin_signal:
TSI0_CH5/PTC17/LLWU_P1/SPI0_SOUT/I2C1_SCL/UART0_RX/BSM_FRAME/DTM_RX,
direction: INPUT, slew_rate: fast,
    drive_strength: low, pull_select: up, pull_enable: enable}
 * BE CAREFUL MODIFYING THIS COMMENT - IT IS YAML SETTINGS FOR THE PINS TOOL
***
 */

/*FUNCTION***********************************************************
*******
 *
 * Function Name : BOARD_InitPins
 * Description   : Configures pin routing and optionally pin electrical features.
 *

*END****************************************************************
******/
void BOARD_InitPins(void) {
    CLOCK_EnableClock(kCLOCK_PortC);        /* Port C Clock Gate Control: Clock enabled
*/

    const port_pin_config_t portc17_pin46_config = {
    kPORT_PullUp,                         /* Internal pull-up resistor is enabled */
    kPORT_FastSlewRate,                   /* Fast slew rate is configured */
    kPORT_PassiveFilterDisable,           /* Passive filter is disabled */
    kPORT_LowDriveStrength,               /* Low drive strength is configured */
    kPORT_MuxAsGpio,                      /* Pin is configured as PTC17 */
    };
    PORT_SetPinConfig(PORTC, PIN17_IDX, &portc17_pin46_config); /* PORTC17 (pin 46)
is configured as PTC17 */
}
#endif // NTAG_I2C
```

Declaration function *BOARD_InitPins()* should be added in to the *pin_mux.h* file.

```
#ifdef NTAG_I2C
/*!
 * @brief Configures pin routing and optionally pin electrical features.
 *
 */
void BOARD_InitPins(void);
#endif // NTAG_I2C
```

1.1.6.2 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:

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

#ifdef NTAG_I2C
        /* Init pins of NTAG  - only FD pin */
        BOARD_InitPins(); // added for NTAG middleware

        /* Init I2C pins for NTAG communication */
        configure_i2c_pins(BOARD_I2C_INSTANCE); // added for NTAG middleware
#endif // NTAG_I2C
```

Function *BOARD_InitPins() initializes the FD pin and function configure_i2c_pins() initializes the I^{2}C periphery.*

Second is the SW initialization performed by function *HAL_I2C_InitDevice() and NTAG* I^{2}C *handler (ntag_handle) is filled by the NFC_InitDevice() function. The interrupt callback initialization (HAL_ISR_RegisterCallback()) have to be inserted also but must be commented from reason written in the chapter 1.1.6.1*

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);
//    HAL_ISR_RegisterCallback((ISR_SOURCE_T)0, ISR_LEVEL_LO, NULL, NULL);
#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
```

1.1.6.3 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
```

app.c

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

hardware_init.c

```
#include "pin_mux.h"…………// added for NTAG middleware
```

pin_mux.c, board.h

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

1.1.6.4 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. Source file *app_ntag.c* consists of the functions which write the BLE pairing NDEF message and default NXP NDEF message. Header file *app_ntag.h* consist of these two NDEF messages.

Here is the printout of the *app_ntag.c* source file:

```
/************************************************************************
*************
* Includes
*************************************************************************
************/
#include "ntag_driver_intern.h"
#include "ntag_defines.h"
#include "ntag_driver.h"
#include "nfc_device.h"
#include "app_ntag.h"

/************************************************************************
*************
* Public functions
*************************************************************************
************/
void NDEF_pairing_write(void)
{
    /* reset default eeprom memory values (smart poster) */
    NFC_WriteBytes(ntag_handle, NTAG_MEM_ADRR_I2C_ADDRESS,
BLE_pairing_NDEF_msg, BLE_pairing_NDEF_msg_Length);

    /* reset pages 56,57,58 */
    NFC_WriteBlock(ntag_handle, 56, Default_Page_56, NTAG_I2C_BLOCK_SIZE);
    NFC_WriteBlock(ntag_handle, 57, Default_Page_57, NTAG_I2C_BLOCK_SIZE);
    NFC_WriteBlock(ntag_handle, 58, Default_Page_58, NTAG_I2C_BLOCK_SIZE);
}


void NDEF_Defaul_write(void)
{
    /* reset default eeprom memory values (smart poster) */
    NFC_WriteBytes(ntag_handle, NTAG_MEM_ADRR_I2C_ADDRESS,
Default_BeginingOfMemory, Default_BeginingOfMemory_length);

    /* reset pages 56,57,58 */
    NFC_WriteBlock(ntag_handle, 56, Default_Page_56, NTAG_I2C_BLOCK_SIZE);
    NFC_WriteBlock(ntag_handle, 57, Default_Page_57, NTAG_I2C_BLOCK_SIZE);
    NFC_WriteBlock(ntag_handle, 58, Default_Page_58, NTAG_I2C_BLOCK_SIZE);
}

/*!
************************************************************************
*********
* @}   end of file
************************************************************************
********** */
```

Here is the printout of the *app_ntag.h* header file:

```
/***********************************************************************
*************
* Include
************************************************************************
************/
#include "ntag_defines.h"
#include "ntag_driver.h"
#include "nfc_device.h"

/*!
************************************************************************
*********
* Public type definitions
************************************************************************
********** */
#ifdef I2C_FSL
#define NTAG_I2C_MASTER_BASEADDR       I2C1
#define I2C_MASTER_CLK_SRC           I2C1_CLK_SRC
#endif

/*!
************************************************************************
*********
* Public memory declarations
************************************************************************
********** */



/* BLE pairing NDEF message
```

```
 *
 * content has to be multiples of 0x10
 * one block consist of the 0x10 bytes
 */
static const uint8_t BLE_pairing_NDEF_msg[] = {
/* 16 */     0xAA, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0xE1, 0x10, 0x6D, 0x00,
/* 2  */       0x03, 0x36,
/* 4  */       0xDA, 0x20, 0x11, 0x01,
/* 32 */
     'a','p','p','l','i','c','a','t','i','o','n','/','v','n','d','.','b','l','u','e','t','o','o','t','h','.','e','p','.','o','o',
'b',
/* 1  */       0x30,
/* 2  */       0x11, 0x00,
/* 6  */       0x04, 0x00, 0x00, 0x9F, 0x04, 0x00,
/* 2  */       0x08, 0x09,
/* 7 */            'F','S','L','_','H','I','D',
/* 1  */       0xFE,
/* -- 73 --  */
     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
/* -- 80 --  this is 5x block */
     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00,
     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00,
     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00,
     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00};

/*Default NDEF message: SmartPoster
 * Title: NTAG I2C Explorer
 * Link: http://www.nxp.com/demoboard/OM5569
 */
static const uint8_t Default_BeginingOfMemory[] = {
     0xAA, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xE1, 0x10,
0x6D, 0x00,
     0x03, 0x5F, 0x91, 0x02, 0x35, 0x53, 0x70, 0x91, 0x01, 0x14, 0x54, 0x02, 0x65, 0x6E,
0x4E, 0x54,
     0x41, 0x47, 0x20, 0x49, 0x32, 0x43, 0x20, 0x45, 0x58, 0x50, 0x4C, 0x4F, 0x52, 0x45,
0x52, 0x51,
     0x01, 0x19, 0x55, 0x01, 0x6E, 0x78, 0x70, 0x2E, 0x63, 0x6F, 0x6D, 0x2F, 0x64, 0x65,
0x6D, 0x6F,
     0x62, 0x6F, 0x61, 0x72, 0x64, 0x2F, 0x4F, 0x4D, 0x35, 0x35, 0x36, 0x39, 0x54, 0x0F,
0x13, 0x61,
     0x6E, 0x64, 0x72, 0x6F, 0x69, 0x64, 0x2E, 0x63, 0x6F, 0x6D, 0x3A, 0x70, 0x6B, 0x67,
0x63, 0x6F,
     0x6D, 0x2E, 0x6E, 0x78, 0x70, 0x2E, 0x6E, 0x74, 0x61, 0x67, 0x69, 0x32, 0x63, 0x64,
0x65, 0x6D,
     0x6F, 0xFE, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00};

static const uint8_t Null_Block[] = {
     0x00, 0x00, 0x00, 0x00,
     0x00, 0x00, 0x00, 0x00,
     0x00, 0x00, 0x00, 0x00,
     0x00, 0x00, 0x00, 0x00};

static const uint8_t Default_Page_56[] = {
     0x00, 0x00, 0x00, 0x00,
     0x00, 0x00, 0x00, 0x00,
     0x00, 0x00, 0x00, 0x00,
     0x00, 0x00, 0x00, 0xFF};

static const uint8_t Default_Page_57[] = {
     0x00, 0x00, 0x00, 0x00,
     0xFF, 0xFF, 0xFF, 0xFF,
     0x00, 0x00, 0x00, 0x00,
     0x00, 0x00, 0x00, 0x00};

static const uint8_t Default_Page_58[] = {
     0x01, 0x00, 0xF8, 0x48,
     0x08, 0x01, 0x00, 0x00,
     0x00, 0x00, 0x00, 0x00,
     0x00, 0x00, 0x00, 0x00};

static const uint32_t Default_BeginingOfMemory_length =
sizeof(Default_BeginingOfMemory);
static const uint16_t BLE_pairing_NDEF_msg_Length = sizeof(BLE_pairing_NDEF_msg);
// BLE NDEF message
static const uint32_t Null_Block_length = sizeof(Null_Block);
static const uint32_t Default_Page_56_length = sizeof(Default_Page_56);
static const uint32_t Default_Page_57_length = sizeof(Default_Page_57);
static const uint32_t Default_Page_58_length = sizeof(Default_Page_58);

extern NFC_HANDLE_T ntag_handle;            // NTAG

/*!
************************************************************************
*********
```

```
* Public prototypes
***************************************************************************
********** */
void NDEF_pairing_write(void);
void NDEF_Defaul_write(void);

/*!
***************************************************************************
*********
* @}   end of file
***************************************************************************
********** */
```

1.2 BLE Demo Application Extension

The BLE demo application is written in the *app.c* file and whole behavior is handled in this file. This demo was extended with writing the NDEF message to the NTAG I²C chip when the button SW3 is pressed. From this reason only *BleApp_HandleKeys()* function (app.c file) is necessary to change. Following C-code printout shows the content of changed function:

```
void BleApp_HandleKeys(key_event_t events)
  {
  uint32_t timeout = NDEF_OVERWRITE_TIMEOUT;

#ifdef NTAG_I2C

  switch (events)
    {
    case gKBD_EventPressPB1_c:  // short press of SW4
      {
      BleApp_Start();
      boNDEFState = TRUE;     // pairing via NDEF is allowed in case the
                          apk. is running
      break;
      }

    case gKBD_EventPressPB2_c:  // short press of SW3
      {
      if (boNDEFState)
        {
        /* added to copy the pairing NDEF message to NTAG_I2C chip */
        NDEF_pairing_write(); // NTAG
        Led3On(); // green LED is lighting
        /* Start advertising timer */
        TMR_StartLowPowerTimer(mNDEFTimerId,gTmrLowPowerSecondTimer_c,
            TmrSeconds(timeout), NDEFTimerCallback, NULL);
        }
      break;
      }

    case gKBD_EventLongPB1_c:   // long press of SW4
      {
      if (mPeerDeviceId != gInvalidDeviceId_c)
        {
        Gap_Disconnect(mPeerDeviceId);
        boNDEFState = FALSE;
        }
      break;
      }

    case gKBD_EventLongPB2_c:   // long press of SW3
      {
      /* nothing action */
      break;
      }

    default:
      break;
    }
  }
```

1.2.1 NDEF Timer

Within the extension of the BLE demo application there was necessary to create NDEF timer. This one performs the time counter from the moment when SW3 button is pressed. Also the NDEF pairing message is written to NTAG I²C chip immediately when SW3 button is pressed. If the time counter is expired the memory content of the NTAG I²C chip is overwritten by default NXP NDEF message.

For timer creation it is necessary to add following to the app.c file:

Add the declaration of the timer handler

This declaration is placed at the beginning 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 the timer (it means initialize)

There are 3 timers within the BLE demo application uses software timer within

The NDEF timer is necessary allocate in the function *BleApp_Config()* in the app.c file. 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

At the end of the *app.c* file is necessary add the *NDEFTimerCallback()* function. If NDEF timer counter expires timer is stoped, green LED is switched off and default NXP NDEF message is written to the NTAG I²C chip.

```
#ifdef NTAG_I2C
/*!
***********************************************************************
*********
 * \brief       Handles timer callback for writing NDEF messages
 *
 * \param[in]    pParam       Calback parameters.

***********************************************************************
********** */
static void NDEFTimerCallback(void * pParam)
  {

      /* Stop Advertising Timer*/
      TMR_StopTimer(mNDEFTimerId);

      Led3Off(); // green LED off

      NDEF_Defaul_write(); // NTAG
  }

#endif // NTAG_I2C
```

1.3 Security change

The sample project for adding NTAG I^{2}C middleware is **hid_device** and is described in chapter 1.1.4. 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.

Finally there are last two parameters of the *gPairingParameters* structure which are necessary to change. Parameter

.localIoCapabilities = gIoDisplayOnly_c,

Has to be changed to

.localIoCapabilities = gIoNone_c,

And parameter

.leSecureConnectionSupported = TRUE,

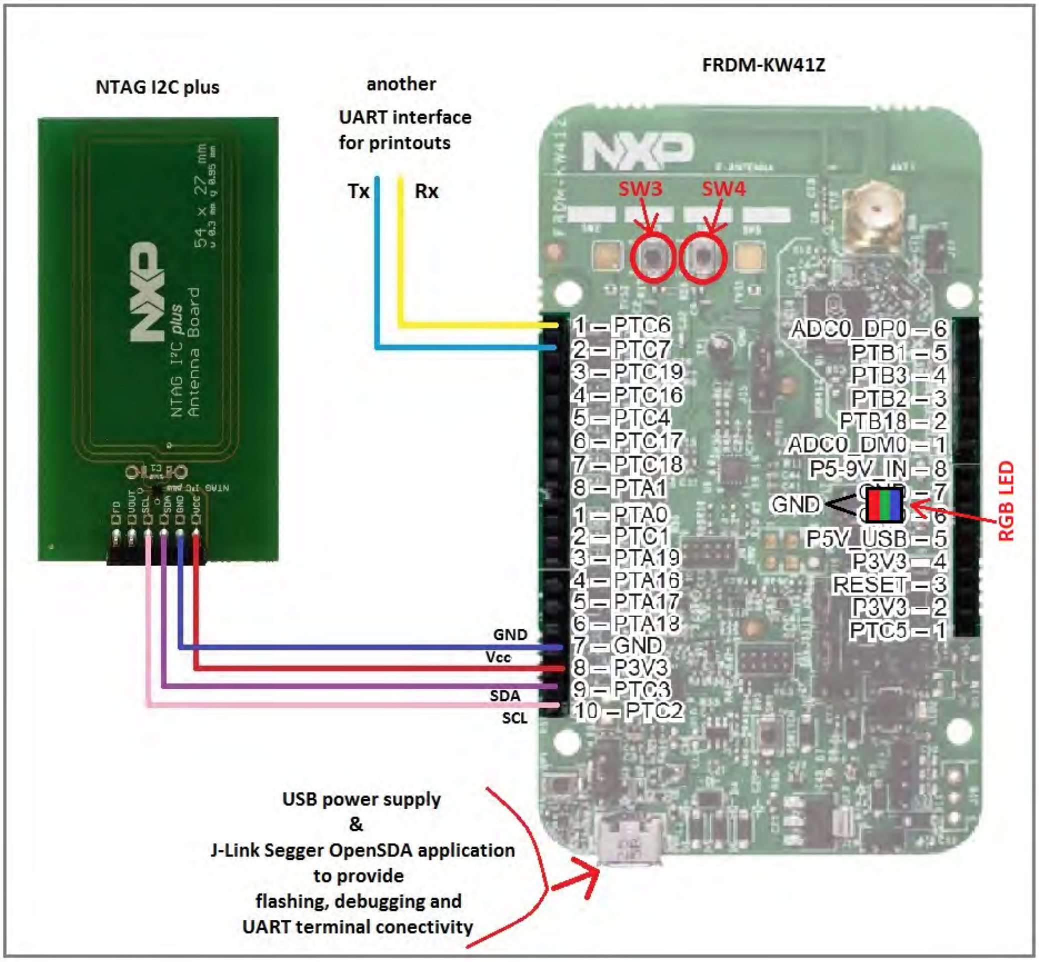
Has to be changed to

.leSecureConnectionSupported = FALSE,

1.4 HW setup

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 BLE demo application 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.



USB power supply
&
J-Link Segger OpenSDA application
to provide
flashing, debugging and
UART terminal conectivity

Table 1: HW configuration within the integration of the NTAG I^{2}C plus PCB board

signal name	description	GPIO pin No.	GPIO direction	specification
I2C – SCL	Interface clock	PTC 2	I^{2}C specific.	open drain SCL
I2C – SDA	Interface data	PTC 3	I^{2}C specific.	open drain SDA
FD	Field detection	PTC 17	input	pull-up
GND	Ground	GND	output	-
VCC_SW	NTAG I^{2}C 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	Button 3 mounted on the FRDM-KW41Z	PTC 4	input	Default SDK configuration
SW4	Button 4 mounted on the FRDM-KW41Z	PTC 5	input	Default SDK configuration

NOTE: **FD** – there is inserted the FD pin description, however this pin and signal is not used within the BLE demo application. The feature of the NFC field detection is not required for overall standard communication between NTAG I^{2}C chip and the MCU.

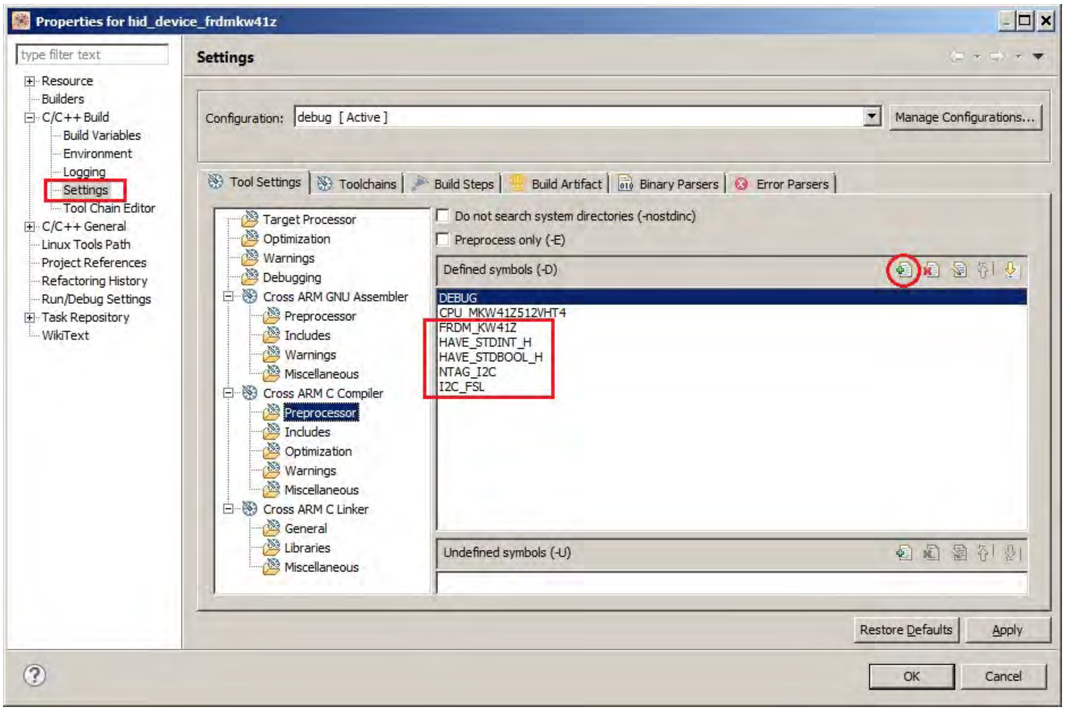
2 BLE Project Properties

2.1 Symbols

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

- FRDM_KW41Z
- NTAG_I2C
- I2C_FSL
- HAVE_STDINT_H
- HAVE_STDBOOL_H

These symbols are conditional defines for compiler and allows using of the NTAG I²C 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.



Properties for hid_device_frdmkw41z

Settings

Configuration: debug [Active]

□ Do not search system directories (-nostdinc)

□ Preprocess only (-E)

Defined symbols (-D)

```
DEBUG
CPU_MKW41Z512VHT4
FRDM_KW41Z
HAVE_STDINT_H
HAVE_STDBOOL_H
NTAG_I2C
I2C_FSL
```

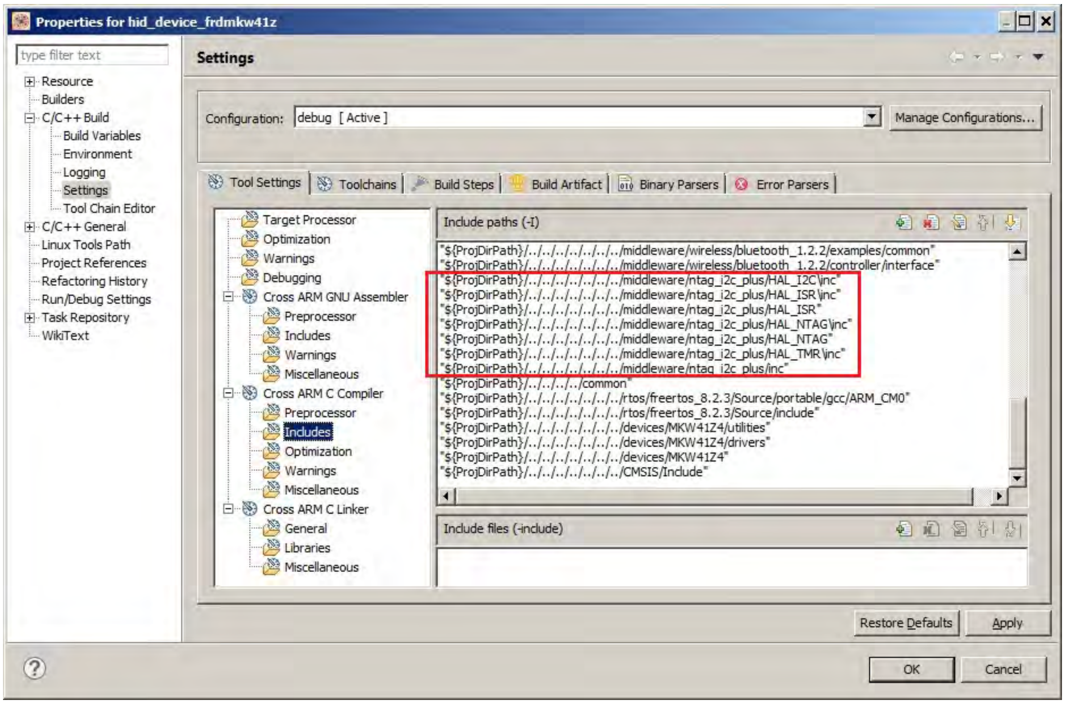
Undefined symbols (-U)

2.2 Include paths

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

- "${ProjDirPath}/../../../../../../../middleware/ntag_i2c_plus/HAL_I2C\inc"
- "${ProjDirPath}/../../../../../../../middleware/ntag_i2c_plus/HAL_ISR\inc"
- "${ProjDirPath}/../../../../../../../middleware/ntag_i2c_plus/HAL_ISR"
- "${ProjDirPath}/../../../../../../../middleware/ntag_i2c_plus/HAL_NTAG\inc"
- "${ProjDirPath}/../../../../../../../middleware/ntag_i2c_plus/HAL_NTAG"
- "${ProjDirPath}/../../../../../../../middleware/ntag_i2c_plus/HAL_TMR\inc"
- "${ProjDirPath}/../../../../../../../middleware/ntag_i2c_plus/inc"

These includes represent the paths which point to source files of the NTAG I²C middleware. Following picture shows the place where the includes are located within the project properties.



Configuration: debug [Active]

Include paths (-I)

```
"${ProjDirPath}/../../../../../../middleware/wireless/bluetooth_1.2.2/examples/common"
"${ProjDirPath}/../../../../../../middleware/wireless/bluetooth_1.2.2/controller/interface"
"${ProjDirPath}/../../../../../../middleware/ntag_i2c_plus/HAL_I2C\inc"
"${ProjDirPath}/../../../../../../middleware/ntag_i2c_plus/HAL_ISR\inc"
"${ProjDirPath}/../../../../../../middleware/ntag_i2c_plus/HAL_ISR"
"${ProjDirPath}/../../../../../../middleware/ntag_i2c_plus/HAL_NTAG\inc"
"${ProjDirPath}/../../../../../../middleware/ntag_i2c_plus/HAL_NTAG"
"${ProjDirPath}/../../../../../../middleware/ntag_i2c_plus/HAL_TMR\inc"
"${ProjDirPath}/../../../../../../middleware/ntag_i2c_plus/inc"
"${ProjDirPath}/../../../../common"
"${ProjDirPath}/../../../../../../rtos/freertos_8.2.3/Source/portable/gcc/ARM_CM0"
"${ProjDirPath}/../../../../../../rtos/freertos_8.2.3/Source/include"
"${ProjDirPath}/../../../../../../devices/MKW41Z4/utilities"
"${ProjDirPath}/../../../../../../devices/MKW41Z4/drivers"
"${ProjDirPath}/../../../../../../devices/MKW41Z4"
"${ProjDirPath}/../../../../../../CMSIS/Include"
```

3 BLE Demo Application

3.1 Build, Run and Debug the demo

This procedure is available on the NXP website (see following picture). For more information see the chapter 1.

How to build and run the BLE demo applications is described on the NXP web site for FRDM-KW41Z development kit [1.] within the third step of the “Getting Started” tab.

3.2 Behavior of the demo

Following steps describes behavior of the changed BLE demo application.

- Standard beginning state for “debug” or “release” mode is that white LED is blinking. This is the idle mode.
- After pressing SW4 button only red LED is blinking. Application goes from idle to searching mode.
- After pressing SW3 button the green LED is lighting and the BLE pairing NDEF is written to the NTAG chip. In this moment it’s able to use NFC feature of the Android phone for Bluetooth pairing.
  NOTE: for successful Bluetooth pairing between the BLE demo application and the Android phone the NFC module and Bluetooth module of the Android phone must be switched on.
- After 10 seconds the green LED is switched off and the pairing NDEF message is overwritten by the default NDEF about the “NTAG I²C plus” demo board. From

this moment it is not possible to pair devices and is necessary press again the button SW3.

- In case that the pairing was successful the green LED is switched off earlier.
- By pressing SW4 button for more than 3 seconds the Bluetooth paired device info is deleted from FRDM-KW41Z and new device may be paired with FRDM-KW41Z.

4 Abbreviations

Table 1. Abbreviations

Acronym	Description
CMSIS	Cortex Microcontroller Software Interface Standard
FD	Field Detection
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
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/demoboard/OM5569-NT322E

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.

MIFARE — is a trademark of NXP B.V.

I²C-bus — is a trademark of NXP B.V.

7 Contents

Please be aware that important notices concerning this document and the product(s) described herein, have been included in the section 'Legal information'.

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