IoT solution made easy with NFC

Session 2: Bluetooth pairing with the NTAG I²C *plus* kit for Arduino pinout

JORDI JOFRE
NFC READERS
NFC EVERYWHERE
27/07/2017

SECURE CONNECTIONS
FOR A SMARTER WORLD

NFC for easy one-tap pairing

Simple secure pairing with a single tap

Pair your phone faster with Bluetooth devices, without conflicts

Tap your Wi-Fi router to get an instant Wi-Fi connection

Pair wireless accessories to your main unit

NFC Benefits

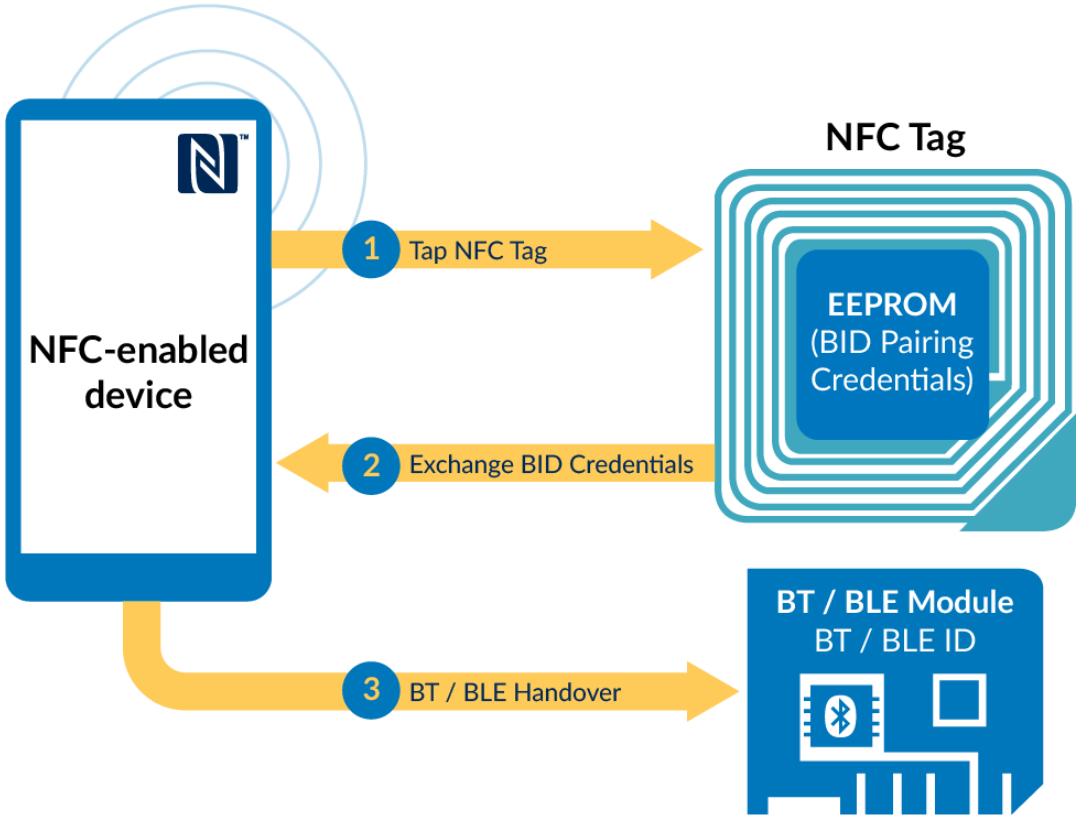
- Pair devices 20x faster than with BLE or Wi-Fi
- Identify a device instantly (no device conflicts or codes)
- Make devices easier to use
- Reduce tech-support costs
- Ensure that accessories are paired to the correct device

Training Mobile Knowledge

How does Bluetooth secure pairing with NFC work?

Easy Bluetooth pairing process

- Tap the NFC tag
- Exchange Pairing Credentials
- Handover to BT / BLE



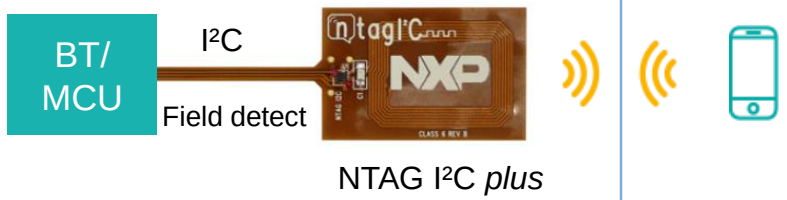
Training Mobile Knowledge

Two solutions to implement secure simple pairing

With NTAG213 label

- Label comes with integrated antenna
- Label can be based anywhere in the product.
- BT MAC address needs to be programmed to both the label and the MCU.

With NTAG I²C *plus*

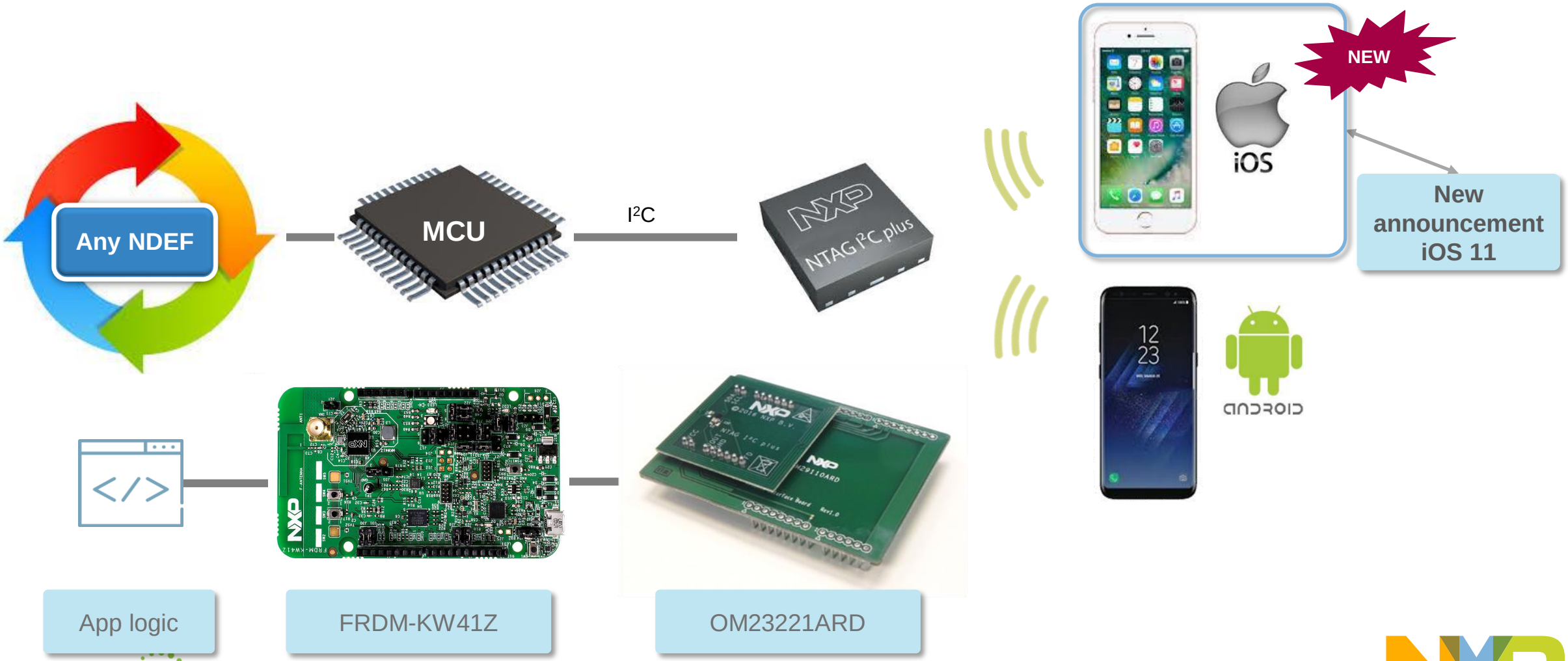


- Pairing target information (BT MAC address) needs to be programmed only once, as MCU and NTAG I²C plus exchange this information.
- Phone can wake up the device through the field detect interrupt.

Today's described solution

Training Mobile Knowledge

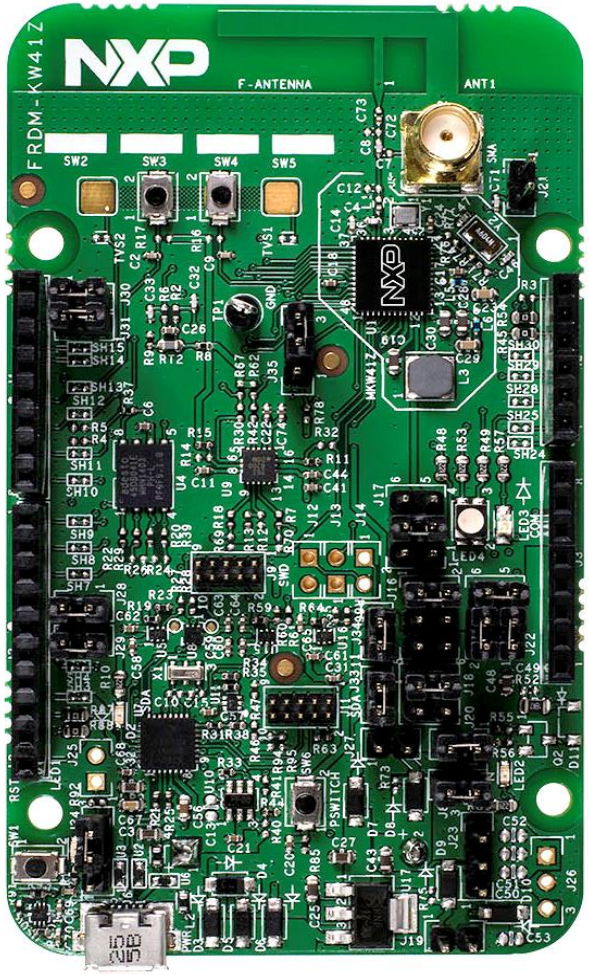
How to initialize Bluetooth pairing with NTAG I²C *plus*



Training Mobile Knowledge

Hardware setup

Kinetis KW41Z/31Z/21Z key differentiators



Multi-Protocol Radio – High performance radio supporting Bluetooth Smart/Bluetooth Low Energy (BLE) v4.2, Generic FSK and IEEE 802.15.4 (Thread) based standards

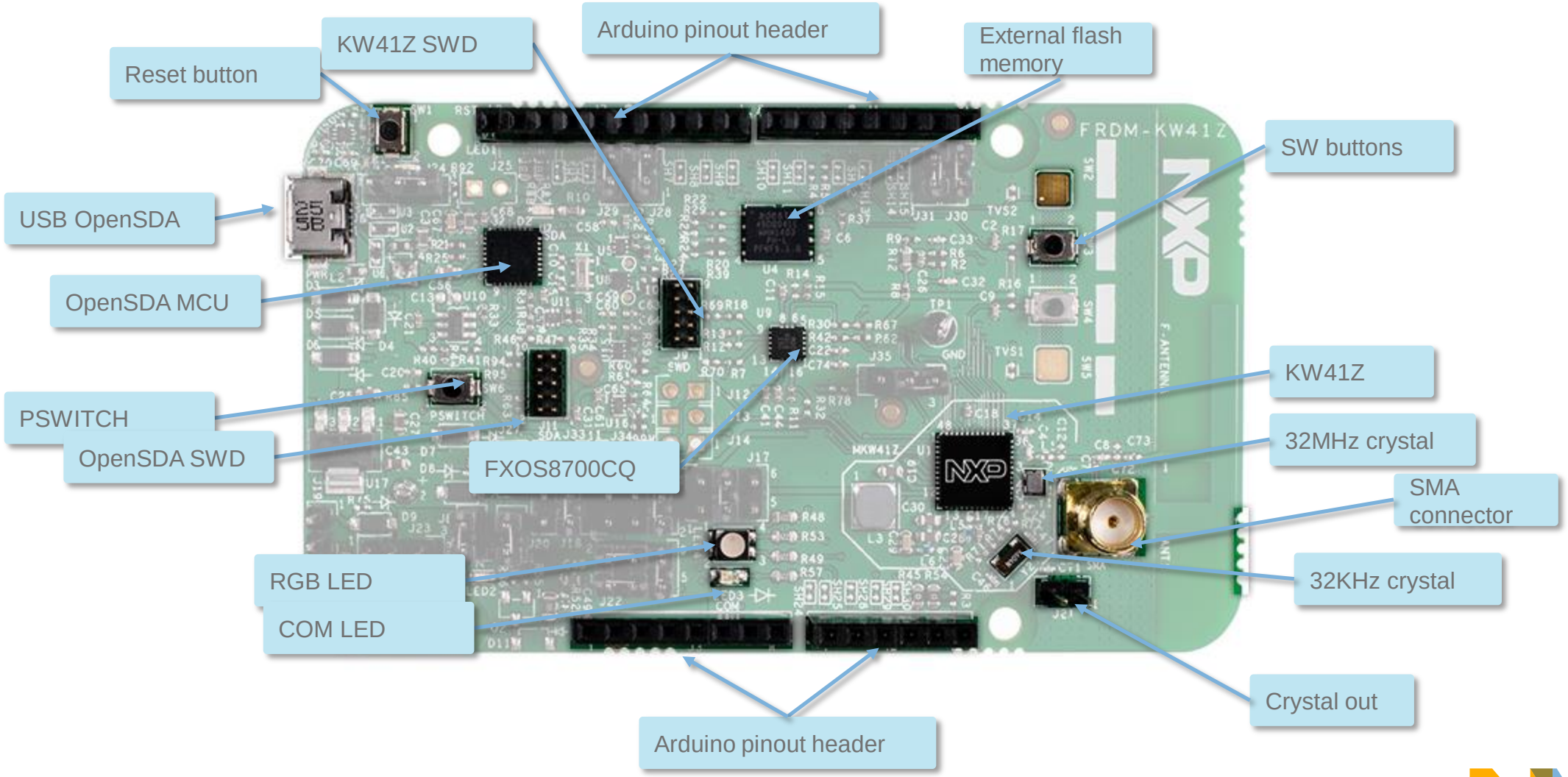
Large Memory – Enough memory to adequately contain desired networking stack(s) with ample room remaining for custom applications

Low Power – Low transmit, receive and standby currents that maximize battery life, including standard coin-cells

Complete Enablement – Fully compliant, qualified Bluetooth Low Energy, Thread and 802.15.4 MAC/PHY. Support for Generic FSK, IPv6 over BLE, SMAC, multiple RTOSes, KSDK 2.0, MCUXPresso IDE and IAR IDEs.

Training Mobile Knowledge

Get to know the FRDM-KW41Z



Training Mobile Knowledge

New NTAG I²C *plus* kit for Arduino pinout

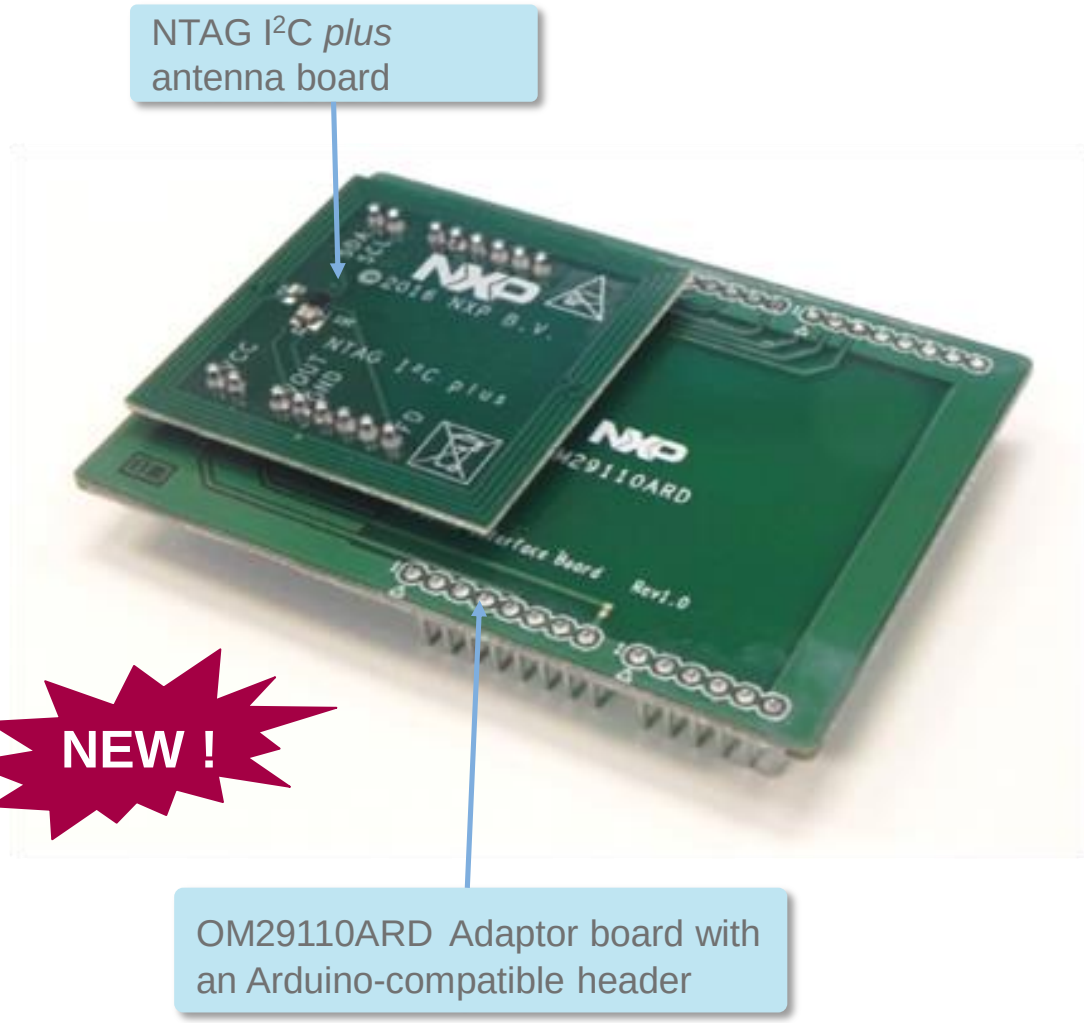
OM23221ARD contents

- NTAG I²C plus PCB antenna board
- Adapter board for Arduino pinout

OM23221ARD features

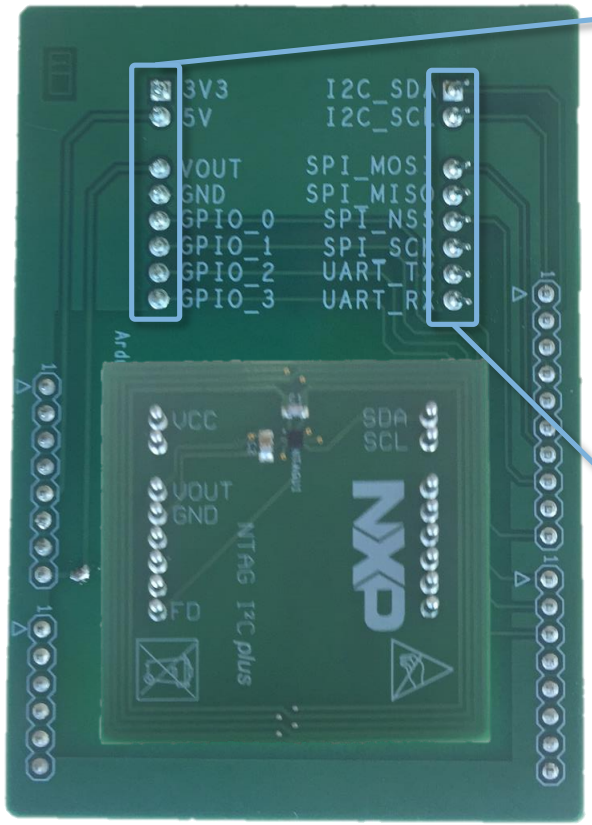
- Connectivity to any device with Arduino pinout, such as NXP Freedom board family (Kinetis) and UDOO Neo (i.MX).
- **Software support for Bluetooth pairing example based on NXP KW41Z**, projects available on Explorer Kit moved to Kinetis platform (e.g. pass-through mechanism) and all examples available through the Kinetis Expert Tool

For additional information please visit:
http://www.nxp.com/demoboard/OM23221ARD



Training Mobile Knowledge

Get to know OM23221ARD board



Pin No.	Pin Name	Purpose
1	3.3V	3.3V supply to the NFC board from host.
2	5V	5V supply to the NFC board from host.
3	Vout	Supply from the NFC board (RF harvesting case)
4	GND	Ground
5	GPIO0	General purpose I/O
6	GPIO1	General purpose I/O
7	GPIO2	General purpose I/O
8	GPIO3	General purpose I/O

Pin No.	Pin Name	Purpose
1	I2C_SDA	I2C data line
2	I2C_SCL	I2C clock line
3	SPI_MOSI	SPI master output, slave input
4	SPI_MISO	SPI master input, slave output
5	SPI_NSS	SPI slave select
6	SPI_SCKI	SPI serial clock
7	UART_TX	Host General purpose I/O pin
8	UART_RX	Host General purpose I/O pin

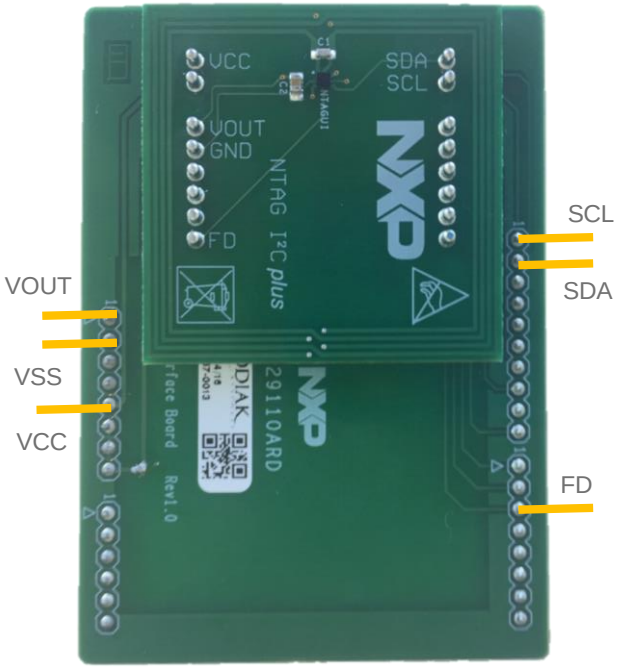
Arduino interface board

It exposes physical interfaces required by the NFC boards. These are:

- Usual power supplies (3.3V-5V)
- Usual IC interfaces (I2C, SPI,UART)
- Generic GPIOs to be used for different purposes (e.g. field detection, interrupt, reset, etc)

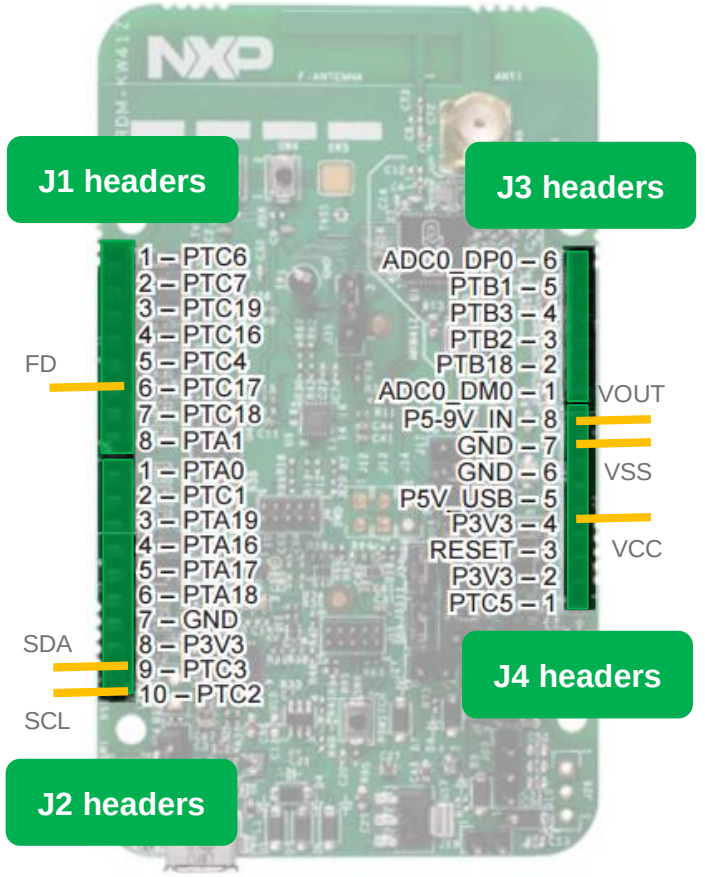
Training Mobile Knowledge

NTAG I²C *plus* wiring with FRDM-KW41Z board



OM29110ARD adaptor board

Pin No.	NTAG I²C *plus* pin	HDR Pin	Connector
1	LA (Antenna connection LA)	-	
2	VSS (GND)	7	J3-GND
3	SCL (Serial Clock I2C)	10	J2-PTC2
4	FD (Field detection)	3	J1-PTC17
5	SDA (Serial data I2C)	9	J2-PTC3
6	VCC ((External power supply)	4	J3-3V3
7	VOUT (Energy Harvesting)	8	J3-P5-9V IN
8	LB (Antenna connection LB)	-	



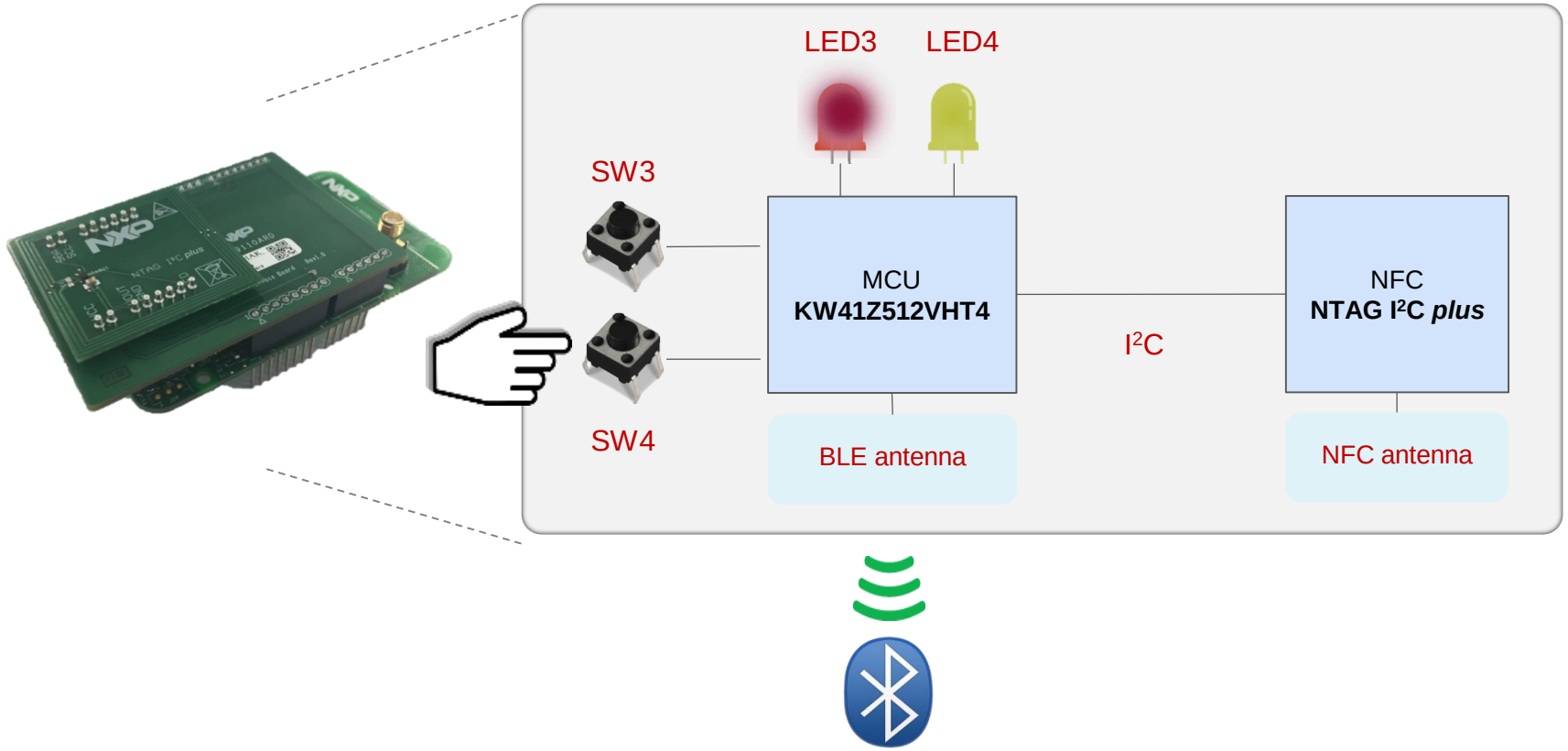
FRDM-KW41Z

Training Mobile Knowledge

BLE pairing with NFC on KW41Z and NTAG I²C *plus*

BLE pairing with NFC on KW41 and NTAG I²C *plus*

Demo application behavior: (1) Application goes from *idle* to *searching* mode



After button **SW4** is pressed:

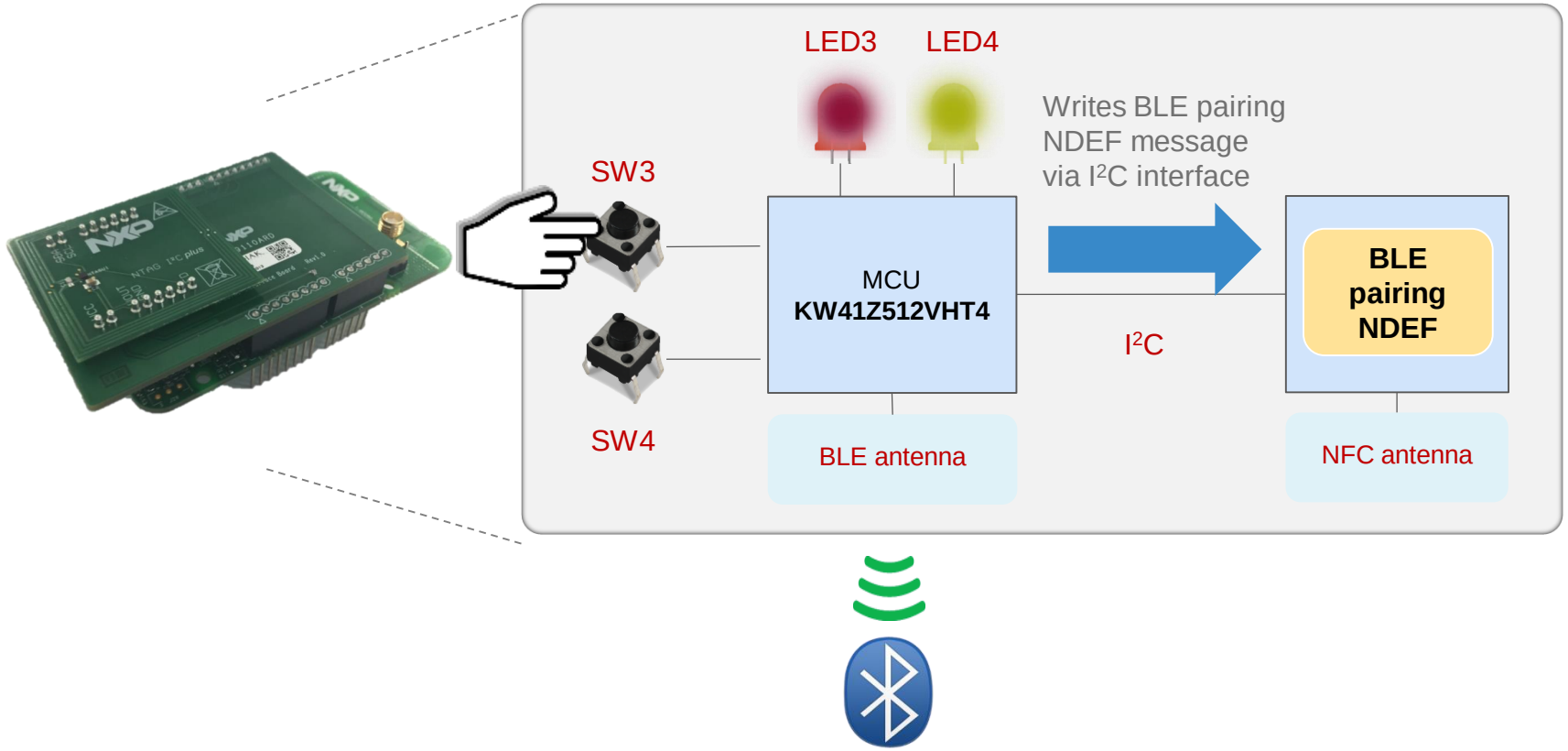
- Application goes from idle to searching mode (BLE active).
- LED 3 blinks.

*Simplified block diagram displaying the components to explain the demo application behavior

Training Mobile Knowledge

BLE pairing with NFC on KW41 and NTAG I²C *plus*

Demo application behavior: (2) Application writes BLE pairing NDEF message



After button **SW3** is pressed:

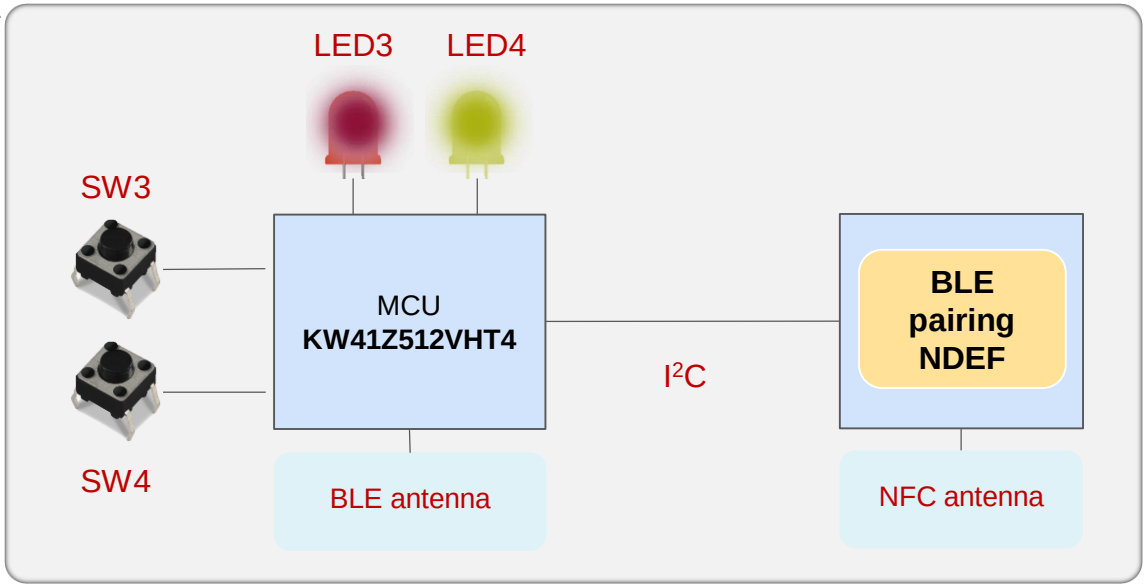
- The BLE pairing NDEF is written to the NTAG I²C *plus* chip.
- LED 4 is set to green.

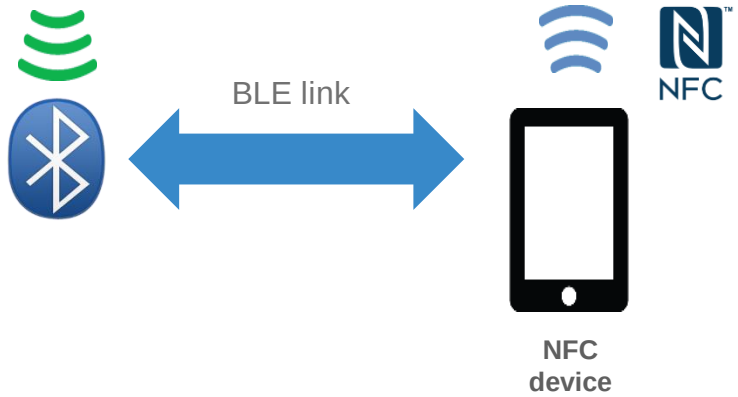
*Simplified block diagram displaying the components to explain the demo application behavior

Training Mobile Knowledge

BLE pairing with NFC on KW41 and NTAG I²C *plus*

Demo application behavior: (3) NFC device reads pairing information and connects





With a tap:

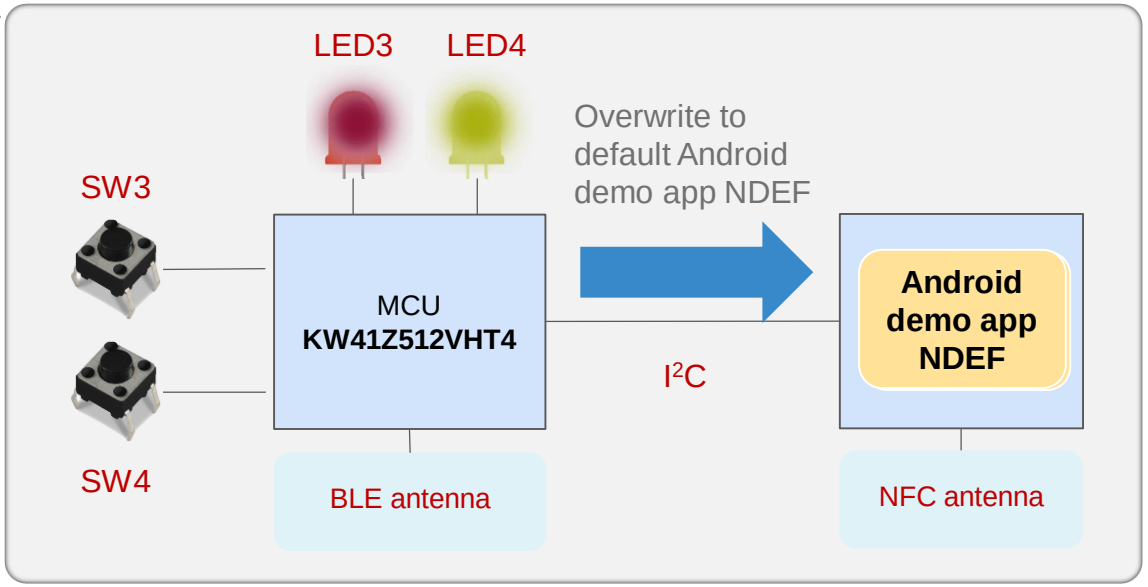
- The NFC device **reads** the BLE pairing NDEF message.
- Automatically established a **BLE connection**.

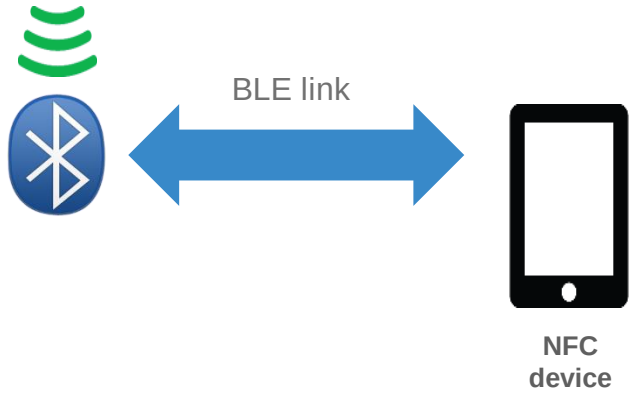
*Simplified block diagram displaying the components to explain the demo application behavior

Training Mobile Knowledge

BLE pairing with NFC on KW41 and NTAG I²C *plus*

Demo application behavior: (4) Application turns back into normal operation mode





After **10 seconds** or **after the NFC device has read** the BLE pairing NDEF message:

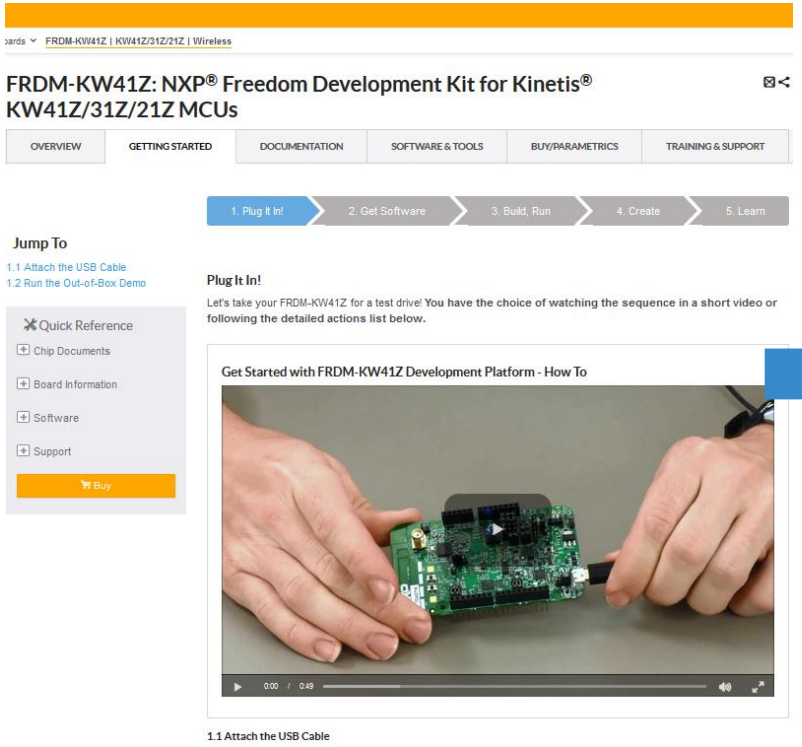
- MCU removes the BLE pairing NDEF
- MCU writes the default NDEF about the NTAG I²C *plus* Android demo app
- LED 4 switches off.

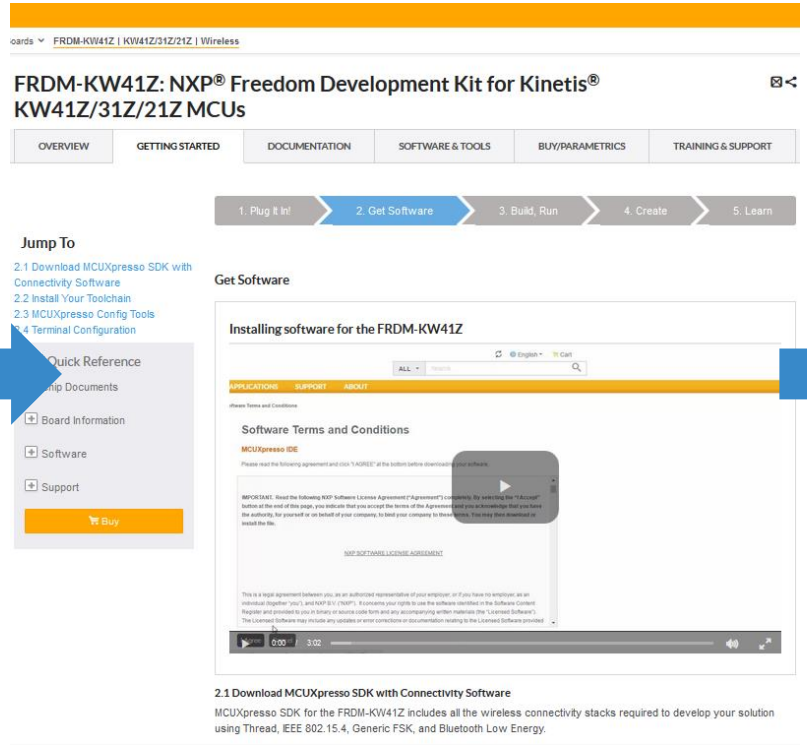
*Simplified block diagram displaying the components to explain the demo application behavior

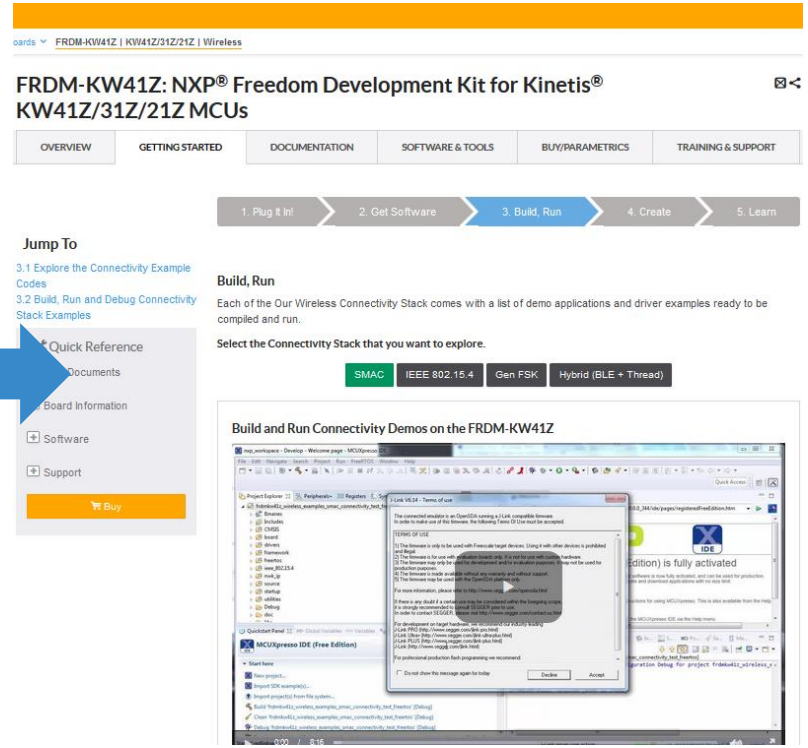
Training Mobile Knowledge

NTAG I²C *plus* integration into FRDM-KW41Z

FRDM-KW41Z startup







Test your FRDM-KW41Z

Download & install MCUXpresso

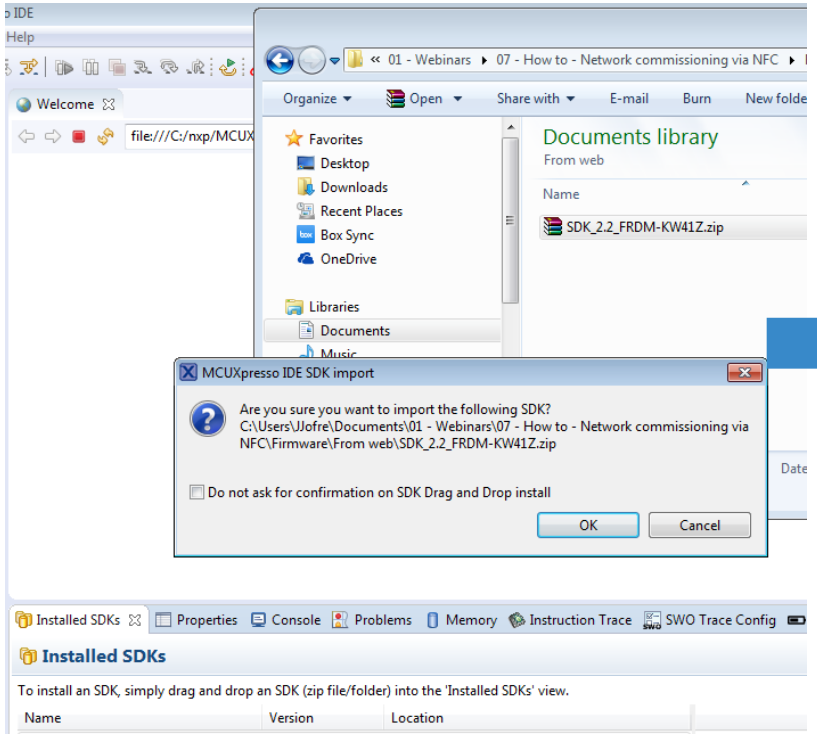
Download & install MCUXpresso SDK for the FRDM-KW41Z

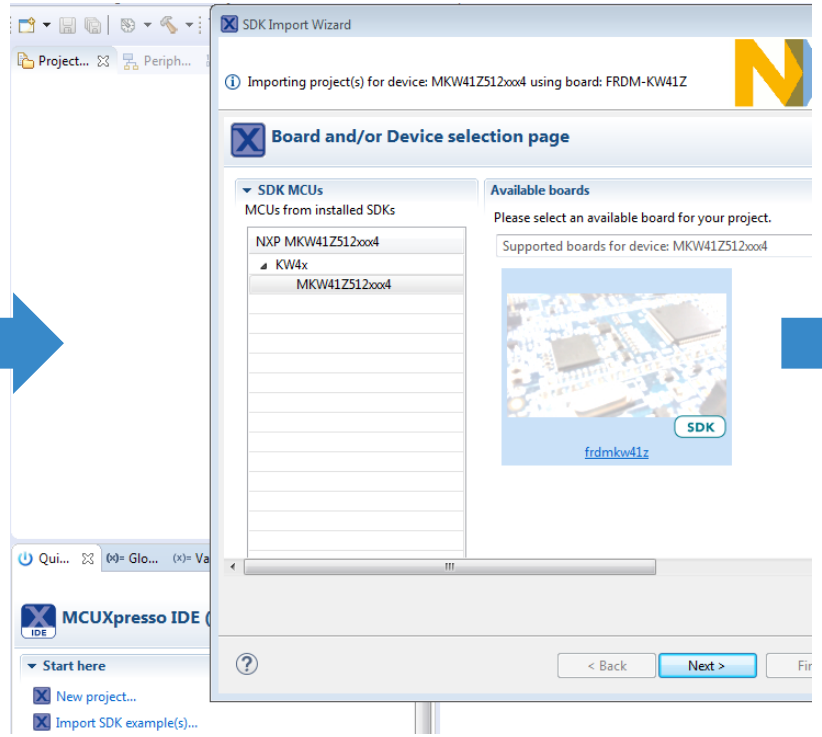
Check & import demo applications included in the MCUXpresso SDK for FRDM-KW41Z in *\boards\frdmkw41z\wireless_examples*

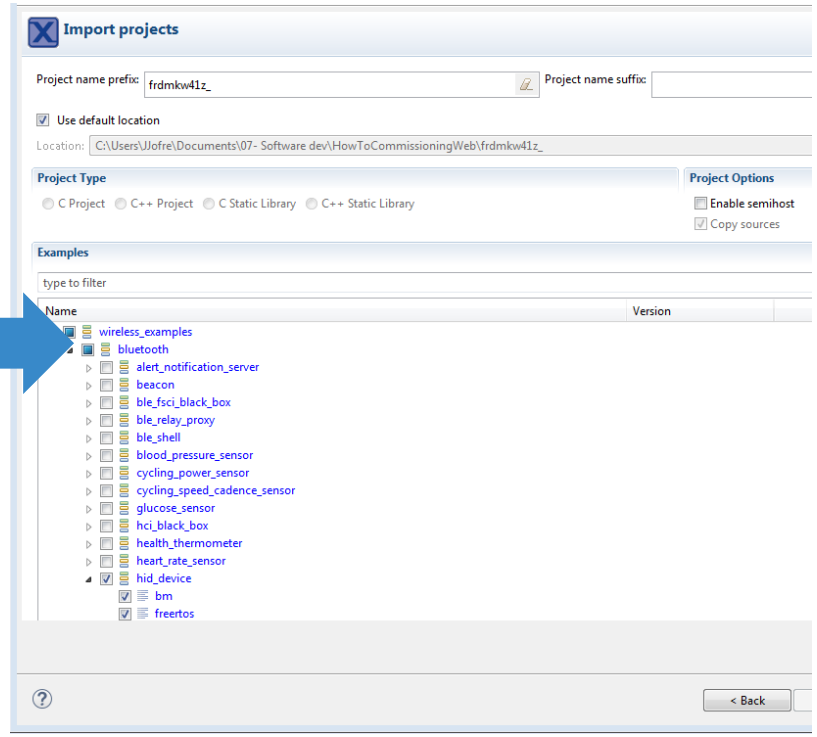
Training Mobile Knowledge

All documentation and tutorials: www.nxp.com/demoboard/FRDM-KW41Z

Importing FRDM-KW41Z SDK example in MCUXpresso toolchain







Import SDK

Import SDK examples

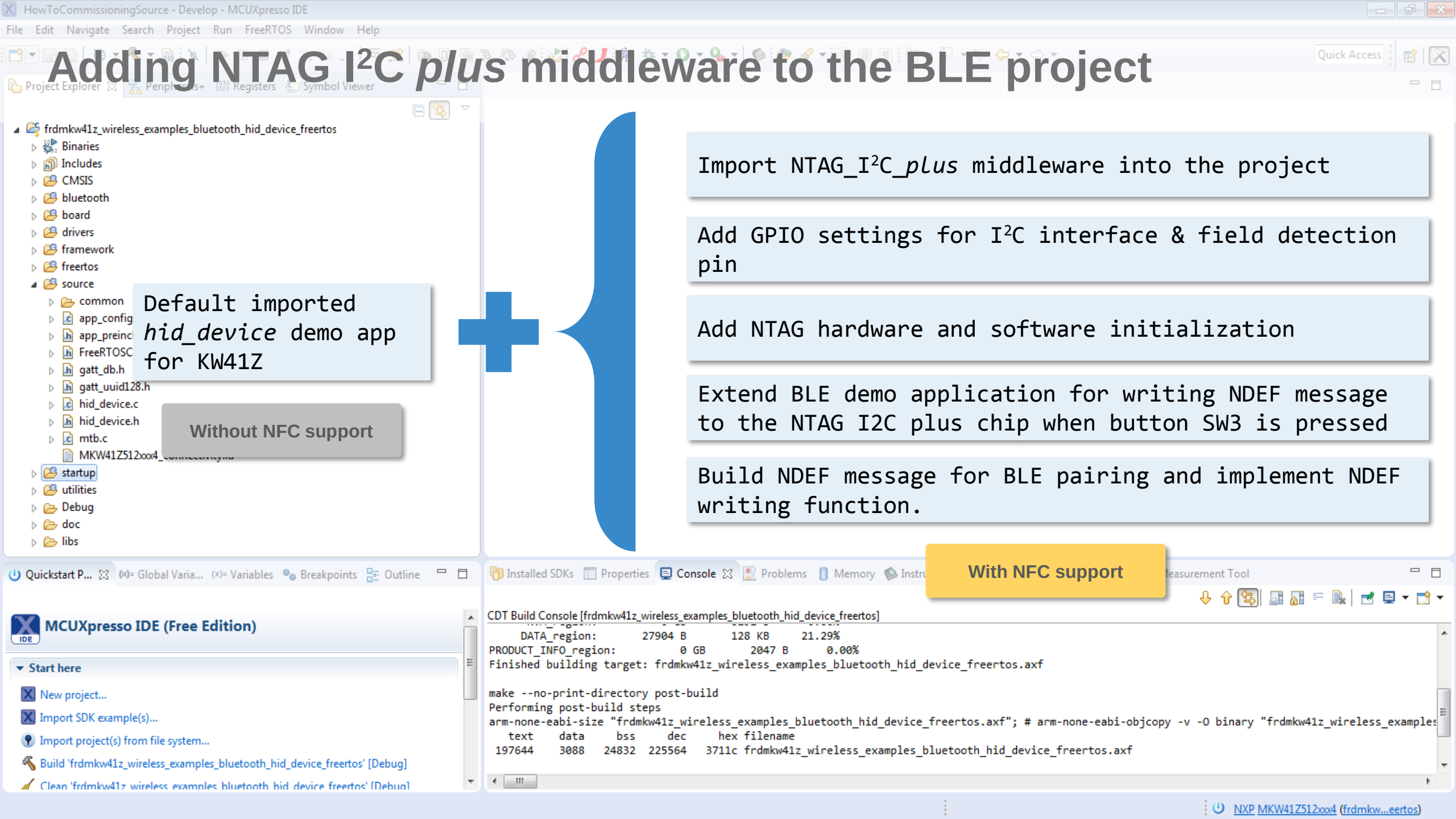
Import */wireless/bluetooth/hid_device* example

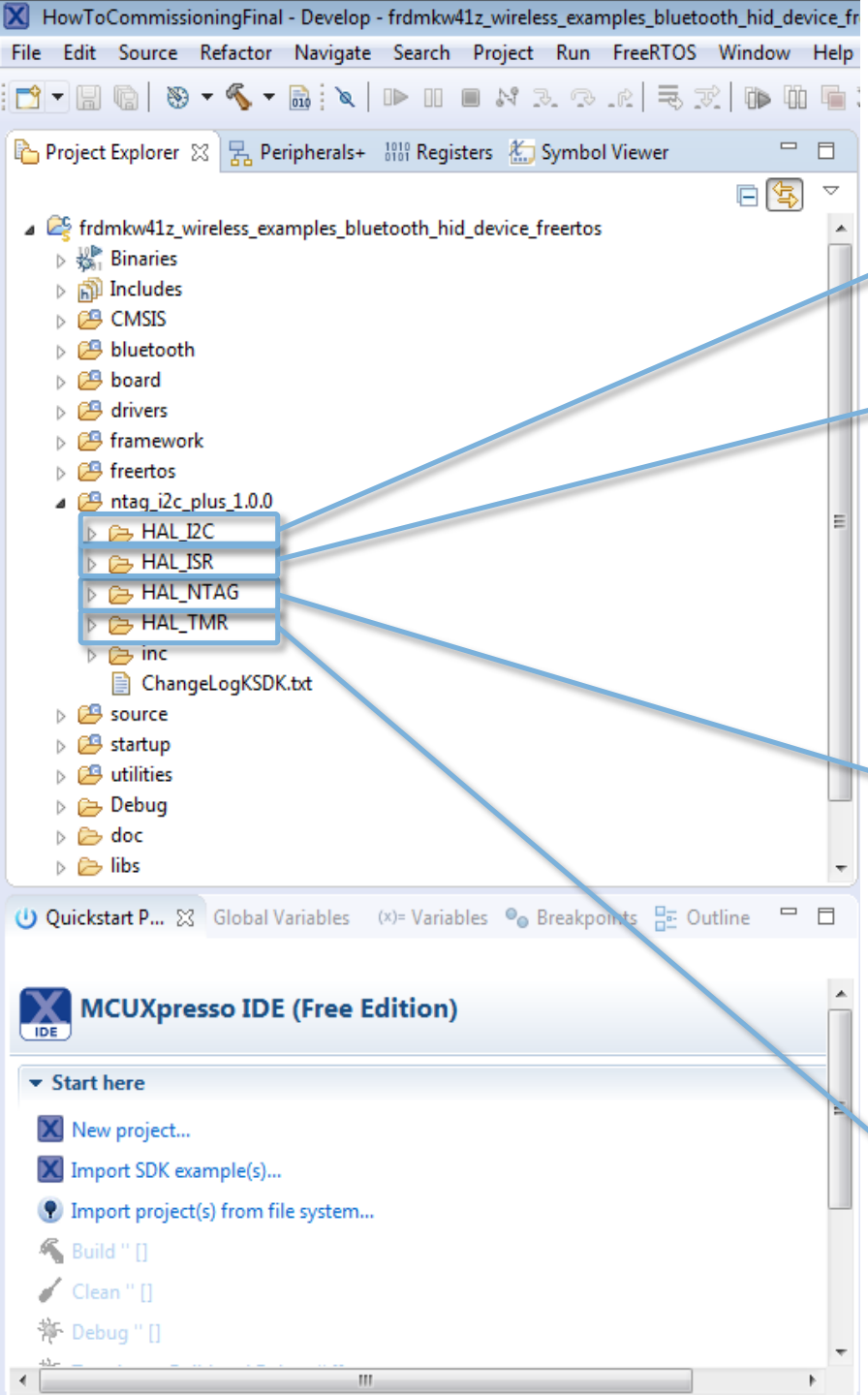
Sample project used as a basis for adding NTAG I²C is **hid_device** located at: *boards\frdmkw41z\wireless_examples\bluetooth\hid_device*

FRDM-KW41Z video tutorials: www.nxp.com/demoboard/FRDM-KW41Z

Training Mobile Knowledge

Adding NTAG I²C *plus* middleware to the BLE project





Importing NTAG I²C *plus* middleware

Interface to access the I²C hardware of the KW41Z.

Interface for registering callbacks and waiting for interrupts of the KW41Z.

Implements the NTAG I²C *plus* command set and offers an API to developers to communicate with NTAG I²C *plus* from the I²C interface.

E.g. HAL_NTAG: Memory operations (I²C side)

```
NTAG_ReadBytes (NTAG_HANDLE_T ntag, uint16_t address,uint8_t *bytes, uint16_t len);
NTAG_WriteBytes(NTAG_HANDLE_T ntag, uint16_t address, const uint8_t *bytes,
uint16_t len);
```

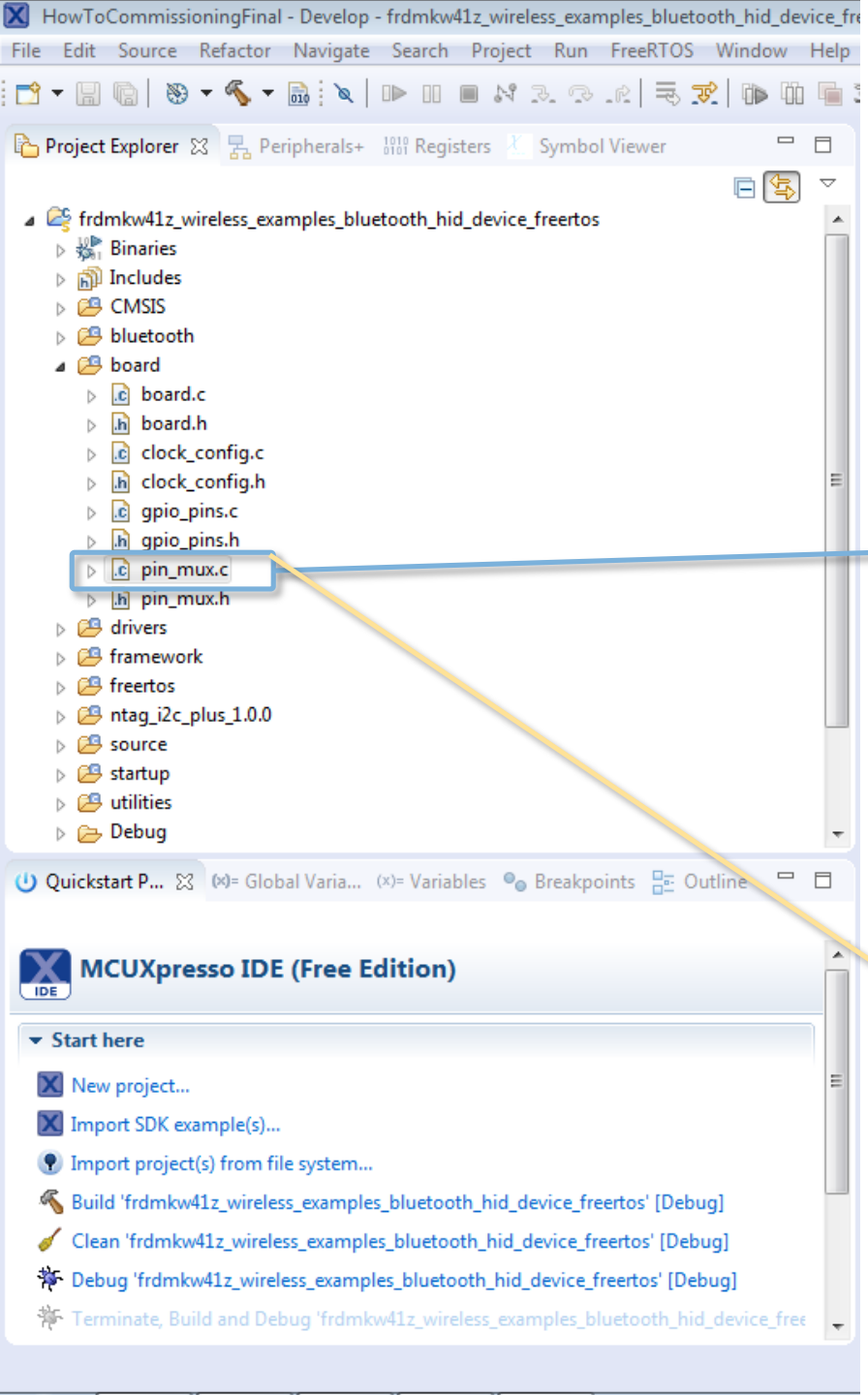
E.g. HAL_NTAG Register operations

```
NTAG_ReadRegister (NTAG_HANDLE_T ntag, uint8_t reg, uint8_t *val);
NTAG_WriteRegister(NTAG_HANDLE_T ntag, uint8_t reg, uint8_t mask, uint8_t val);
```

E.g. HAL_NTAG Setting SRAM for pass-throug mode operation

```
NTAG_SetPthruOnOff(NTAG_HANDLE_T ntag, BOOL on)
NTAG_SetTransferDir(NTAG_HANDLE_T ntag, NTAG_TRANSFER_DIR_T dir)
```

Interface to access the timing hardware of the KW41Z.



GPIO pins settings

I²C pins

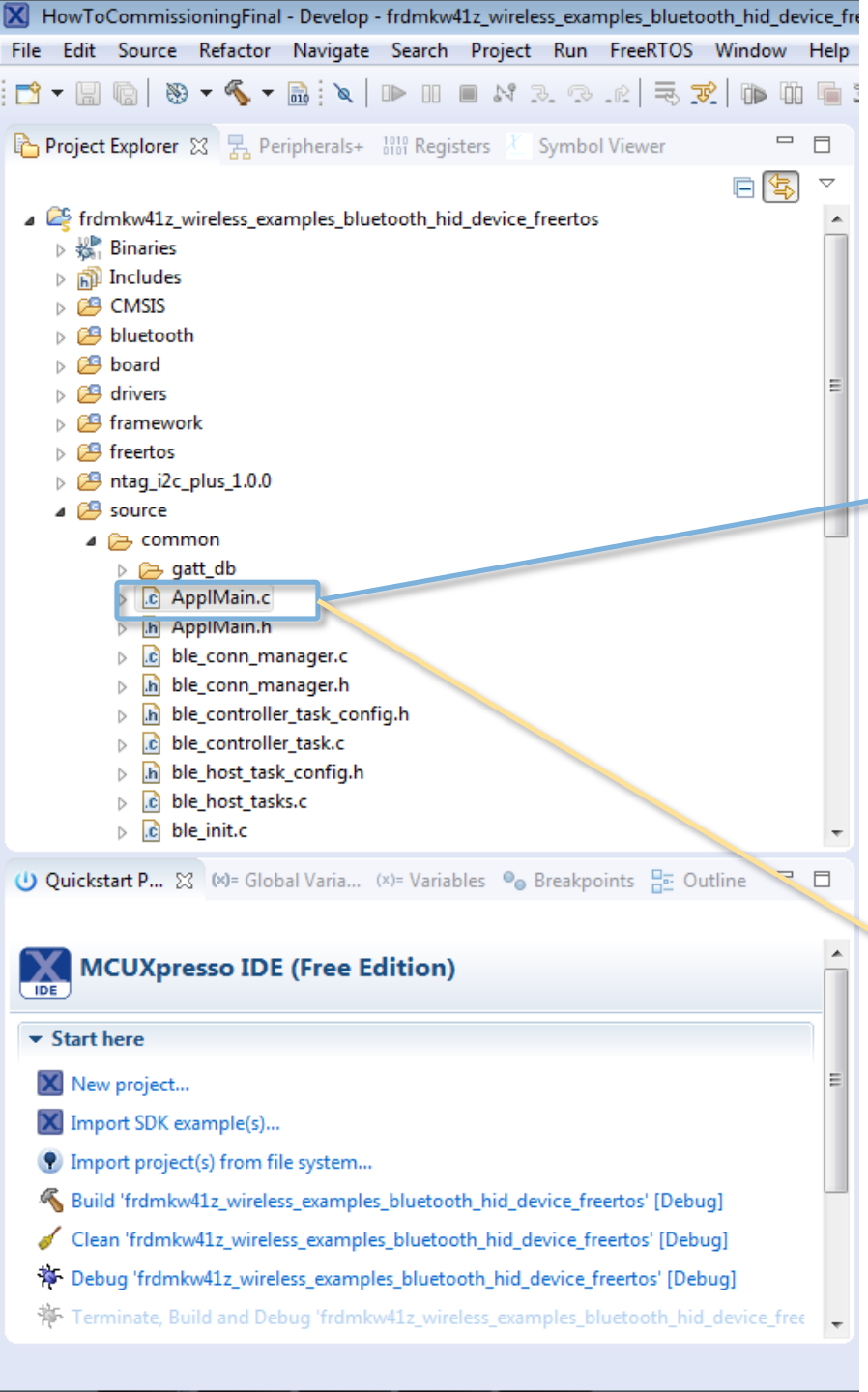
```
void BOARD_InitI2C(void) {
   /* PORTB0 (pin 16) is configured as I2C0_SCL */
   PORT_SetPinMux(PORTB, PIN0_IDX, kPORT_MuxAlt3);
   PORTB->PCR[0] = ((PORTB->PCR[0] & (~(PORT_PCR_PS_MASK | PORT_PCR_PE_MASK |
         PORT_PCR_ISF_MASK)))| PORT_PCR_PS(PCR_PS_UP)| PORT_PCR_PE(PCR_PE_ENABLED));

   /* PORTB1 (pin 17) is configured as I2C0_SDA */
   PORT_SetPinMux(PORTB, PIN1_IDX, kPORT_MuxAlt3);
   PORTB->PCR[1] = ((PORTB->PCR[1] & (~(PORT_PCR_PS_MASK | PORT_PCR_PE_MASK |
         PORT_PCR_ISF_MASK))) | PORT_PCR_PS(PCR_PS_UP)|
PORT_PCR_PE(PCR_PE_ENABLED));
```

FD pin

```
#ifdef NTAG_I2C
   // initialization FD pin
   #define PIN17_IDX17u   /*!< Pin number for pin 17 in a port */
#endif // NTAG_I2C
```

NTAG I²C *plus* SW and HW initialization

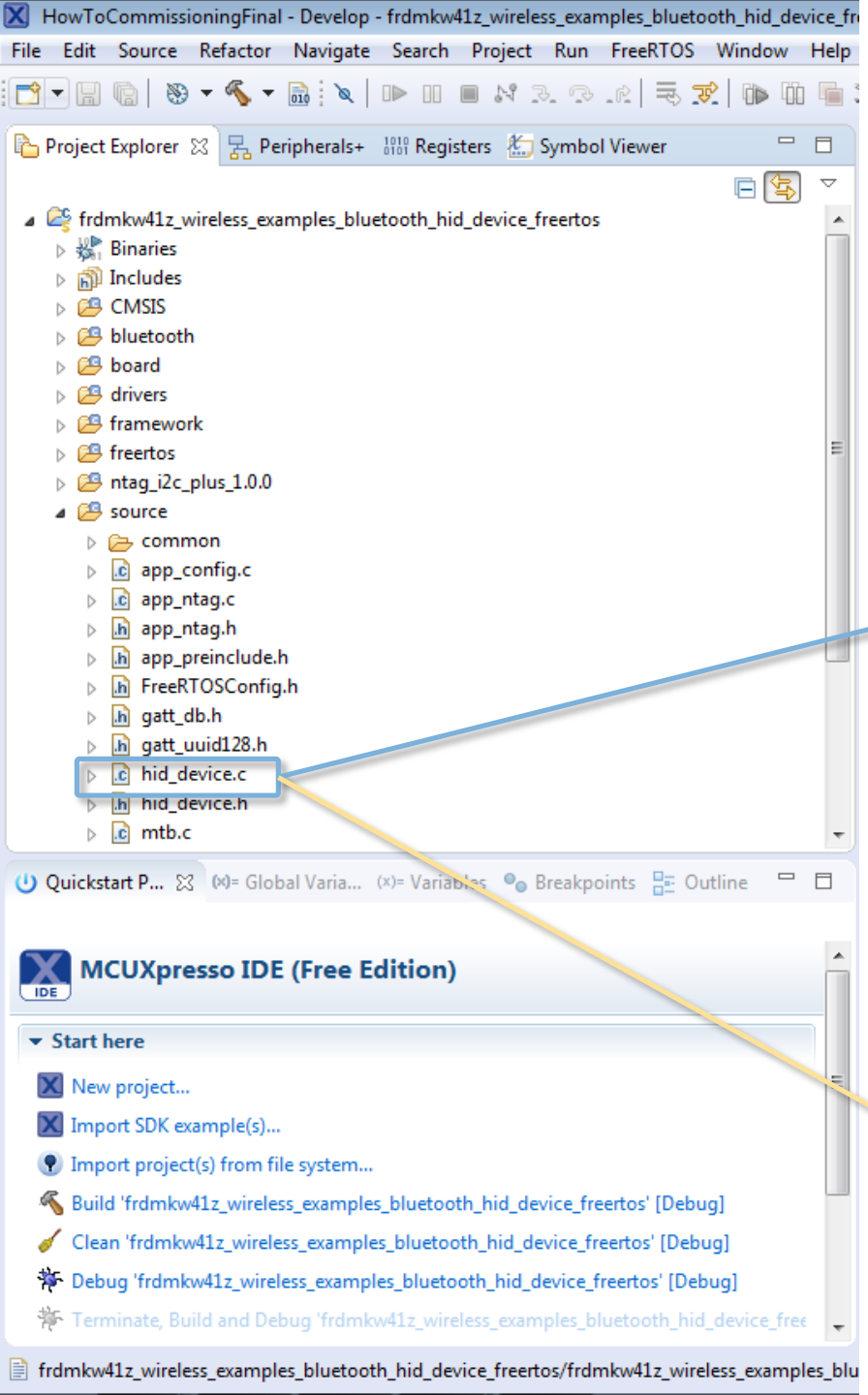



```c
#ifdef NTAG_I2C
   /* NTAG middleware module */
   #include "HAL_I2C_driver.h"
   #include "HAL_I2C_kinetis_fsl.h"
   #include "app_ntag.h"
#endif //NTAG_I2C


#ifdef NTAG_I2C
   NFC_HANDLE_T ntag_handle; // NTAG handle
#endif // NTAG_I2C
```

```c
#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
```



HID_device demo extensions

```
void BleApp_HandleKeys(key_event_t events){
#ifdef NTAG_I2C
   switch (events){
      case gKBD_EventPressPB1_c: {  // short press of SW4
            BleApp_Start();
            boNDEFState = TRUE;    // pairing via NDEF is allowed
            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);}

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

```
void NDEFTimerCallback(void * pParam){
   TMR_StopTimer(mNDEFTimerId); /* Stop Advertising Timer*/
   Led3Off(); // green LED off
   NDEF_Defaul_write(); } // NTAG
```

NTAG I²C *plus* integration into FRDM-KW41Z

NDEF details

Formats for data exchange

NFC data exchange format (NDEF)

- Specifies a common data format for NFC Forum-compliant devices and NFC Forum-compliant tags.
- It is used to describe how a set of actions are to be encoded onto an NFC tag (e.g. open a URL, create an SMS, create an email, etc.).
- The benefit of using NDEF is that you do not need to have custom software running on the touching device.

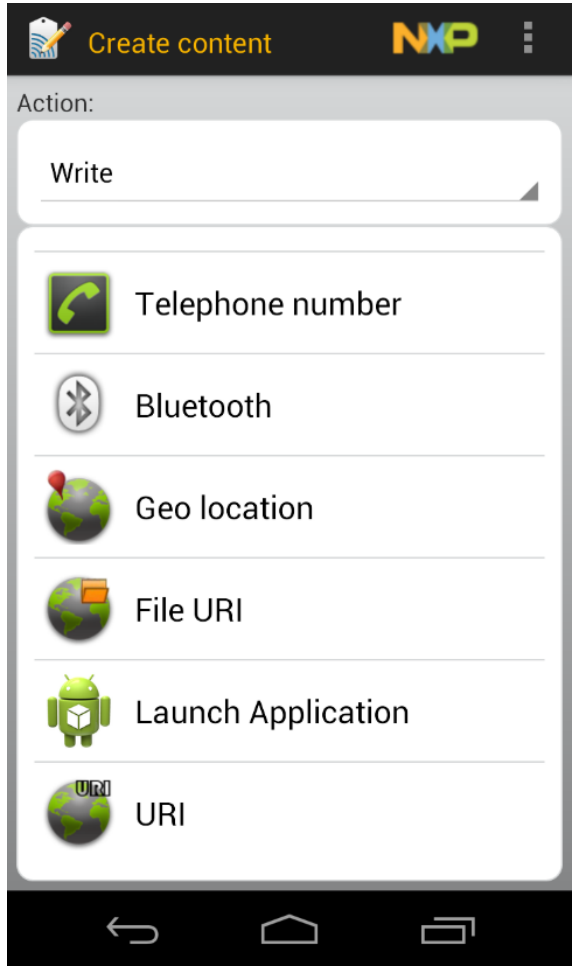
Formats for data exchange

Common NFC record types

- Common NFC record types:
  - **vCard**: Stores contact information (e.g. electronic business cards)
  - **URI:** Stores Universal Resource Identifiers (URIs), which include web addresses and other network resources and files
  - **Text:** Stores text strings in multiple languages.
  - **Smart poster:** Stores text strings, URLs, SMS or phone numbers.
  - **Connection handover:** Stores pairing with Bluetooth, Wi-Fi or other protocols
  - **Device information**: Stores basic details about the device mode and its identity.
  - **Signature:** Provides an algorithm or certificate type for use as a digital signature

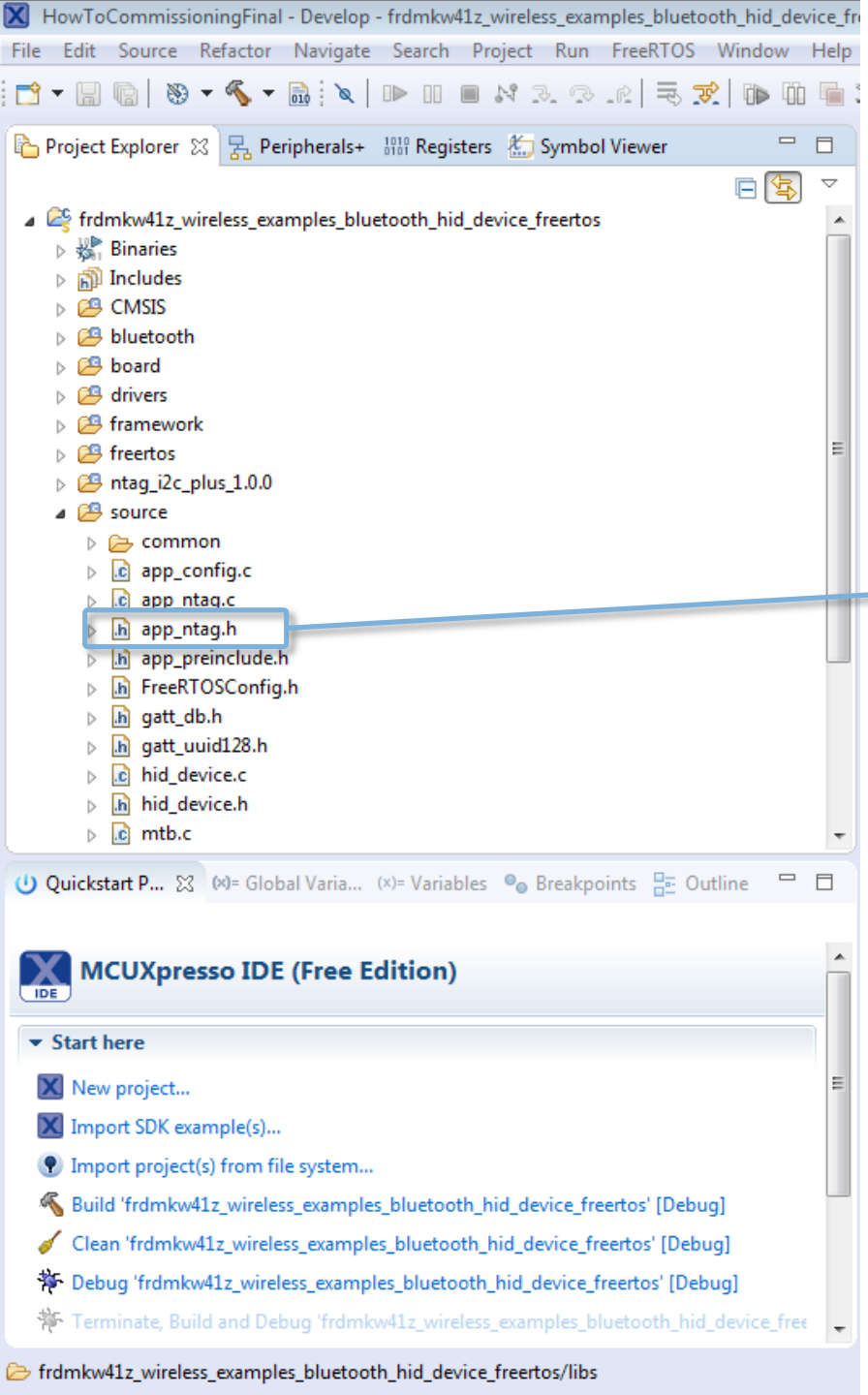
Default NDEF used in the app

Pairing NDEF used in the app



* For more on these formats, check the NFC Forum website (nfc-forum.org)

Training Mobile Knowledge



NDEF message for BLE pairing details

```
static const uint8_t BLE_pairing_NDEF_msg[] = {
  0xAA, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00,
  0xE1, 0x10, 0x6D, 0x00,
  0x03, 0x36, 0xDA, 0x20,
  0x11, 0x01, 0x61, 0x70,
  0x70, 0x6C, 0x69, 0x63,
  0x61, 0x74, 0x69, 0x6F,
  0x6E, 0x2F, 0x76, 0x6E,
  0x64, 0x2E, 0x62, 0x6C,
  0x75, 0x65, 0x74, 0x6F,
  0x6F, 0x74, 0x68, 0x2E,
  0x65, 0x70, 0x2E, 0x6F,
  0x6F, 0x62, 0x30, 0x11,
  0x00, 0x04, 0x00, 0x00,
  0x9F, 0x04, 0x00, 0x08,
  0x09, 0x46, 0x53, 0x4C,
  0x5F, 0x48, 0x49, 0x44,
  0xFE, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00 };
```

Capability Container (Magic Number =0xE1, Version=1.0, Memory size=872bytes, Read/Write access)

TLV (Type=NDEF, Length=54 octets)

Record Type Name: **application/vnd.bluetooth.ep.oob** (ASCII)

MAC address = **00:04:9F:00:00:04** (LSB- 6 bytes)

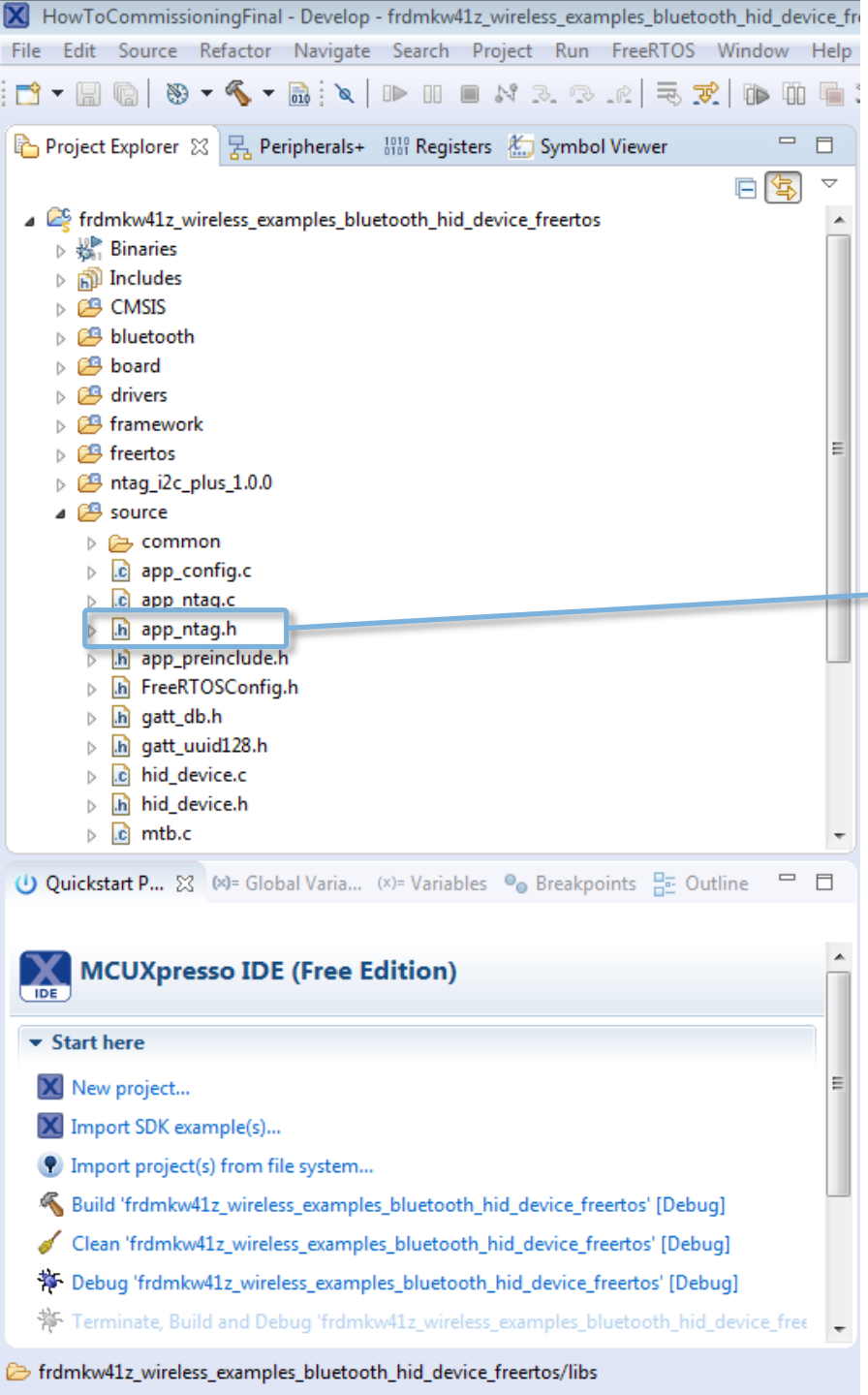
Complete local name = **“FSL_HID”** (ASCII)

Terminator TLV (0xFE) + Padding ‘00’ to get a multiple of 16 octets

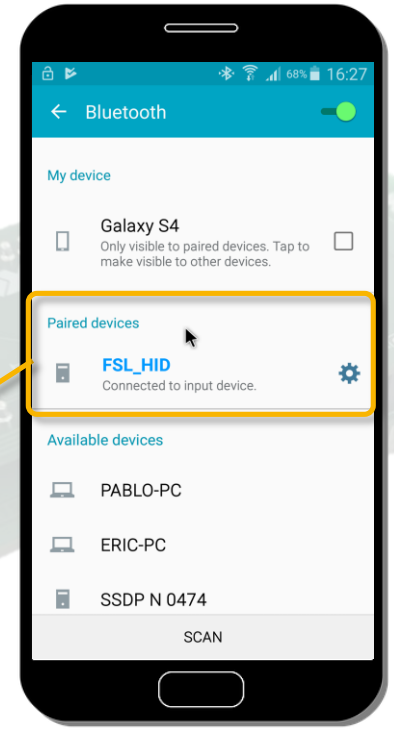
Doc 1: NFC Forum Type 2 Tag Operation Specification
Doc 2: NFC Data Exchange Format (NDEF) Technical Specification
Doc 3: Connection Handover Technical Specification
Doc 4: Bluetooth® Secure Simple Pairing Using NFC

* Data is written in blocks of 16 bytes from the I²C interface but is shown in lines of 4 bytes for reading convenience)

NDEF message for BLE pairing details

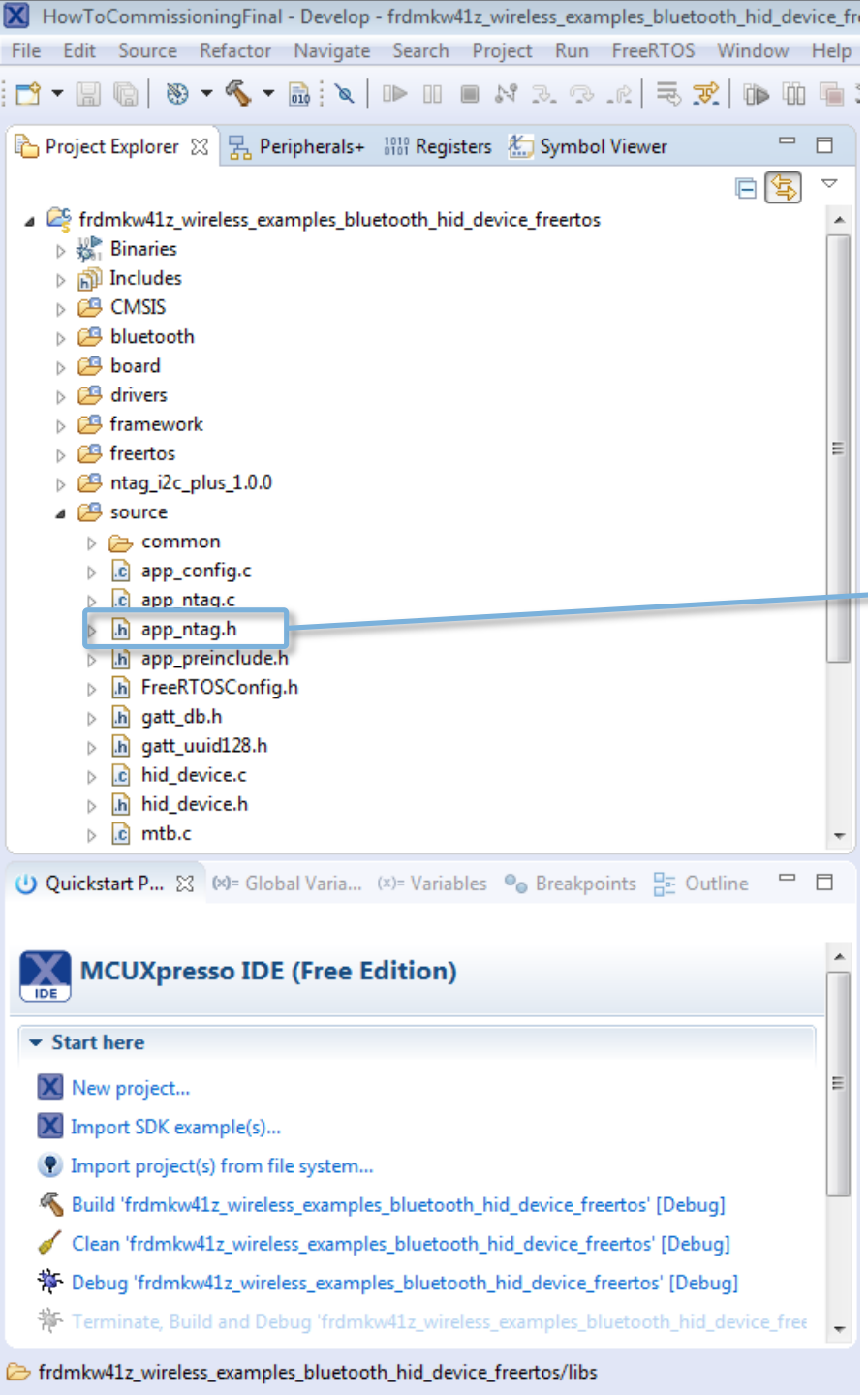



```
static const uint8_t BLE_pairing_NDEF_msg[] = {
  0xAA, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00,
  0xE1, 0x10, 0x6D, 0x00,
  0x03, 0x36, 0xDA, 0x20,
  0x11, 0x01, 0x61, 0x70,
  0x70, 0x6C, 0x69, 0x63,
  0x61, 0x74, 0x69, 0x6F,
  0x6E, 0x2F, 0x76, 0x6E,
  0x64, 0x2E, 0x62, 0x6C,
  0x75, 0x65, 0x74, 0x6F,
  0x6F, 0x74, 0x68, 0x2E,
  0x65, 0x70, 0x2E, 0x6F,
  0x6F, 0x62, 0x30, 0x11,
  0x00, 0x04, 0x00, 0x00,
  0x9F, 0x04, 0x00, 0x08,
  0x09, 0x46, 0x53, 0x4C,
  0x5F, 0x48, 0x49, 0x44,
  0xFE, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00 };
```



Doc 1: NFC Forum Type 2 Tag Operation Specification
Doc 2: NFC Data Exchange Format (NDEF) Technical Specification
Doc 3: Connection Handover Technical Specification
Doc 4: Bluetooth® Secure Simple Pairing Using NFC

* Data is written in blocks of 16 bytes from the I²C interface but is shown in lines of 4 bytes for reading convenience)



Default NDEF message details

```
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};
```

Capability Container (Magic Number =0xE1, Version=1.0, Memory size=872bytes, Read/Write access)

TLV (Type=NDEF, Length=95 octets)

Text record = “**NTAG I2C EXPLORER**”

URI record = **http://www.nxp.com/demoboard/OM5569**

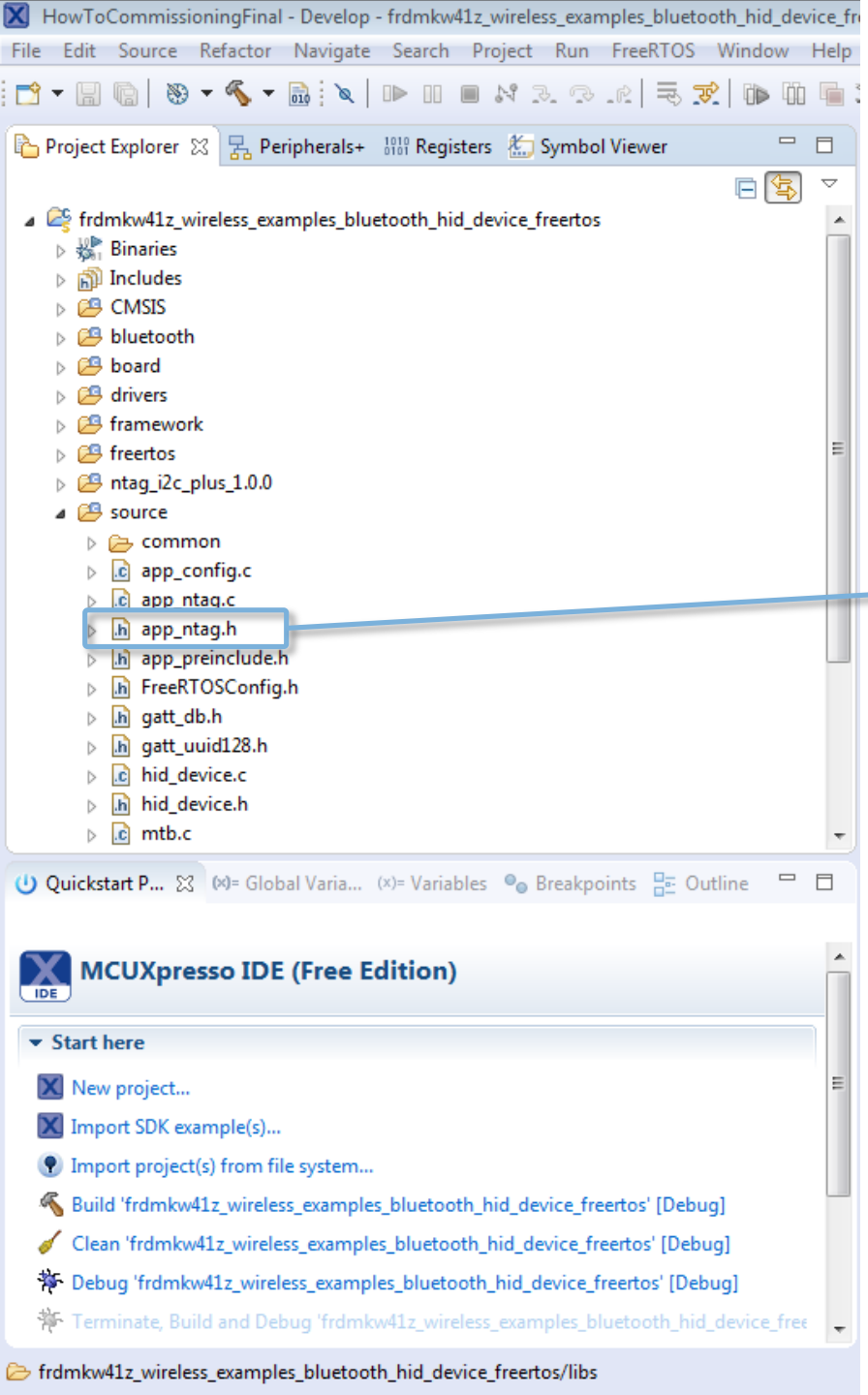
Android application record= **android.com.pkg com.nxp,ntagi2cdemo**

Terminator TLV (0xFE) + Padding ‘00’ to get a multiple of 16 octets

Doc 1: NFC Forum Type 2 Tag Operation Specification
Doc 2: NFC Data Exchange Format (NDEF) Technical Specification
Doc 3: NFC Smart Poster Record Type Definition (RTD)

* Data is written in blocks of 16 bytes from the I²C interface but is shown in lines of 8 bytes for reading convenience)

Default NDEF message details



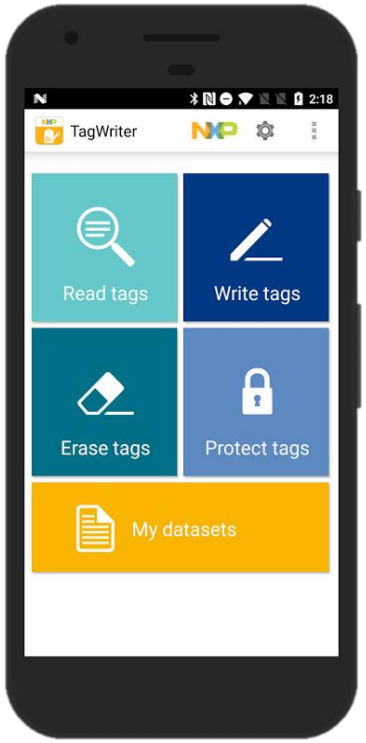

```
static const uint8_t Default_BeginingOfMemory[] = {
  0xAA, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0xE1, 0x10, 0x6D, 0x00,
  0x03, 0x5F, 0x91, 0x02, 0x35, 0x53, 0x70, 0x91,
  0x01, 0x14, 0x54, 0x02, 0x65, 0x6E, 0x4E, 0x54,
  0x41, 0x47, 0x20, 0x49, 0x32, 0x43, 0x20, 0x45,
  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};
```

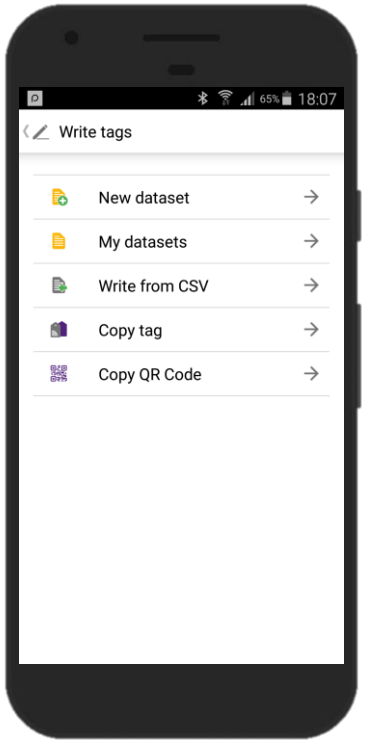
Doc 1: NFC Forum Type 2 Tag Operation Specification
Doc 2: NFC Data Exchange Format (NDEF) Technical Specification
Doc 3: NFC Smart Poster Record Type Definition (RTD)

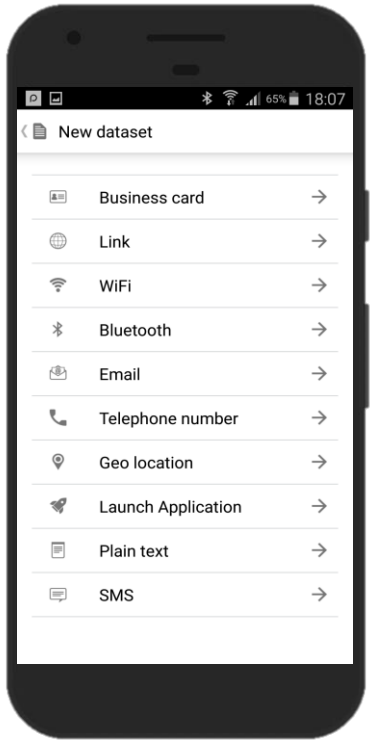
* Data is written in blocks of 16 bytes from the I²C interface but is shown in lines of 8 bytes for reading convenience)

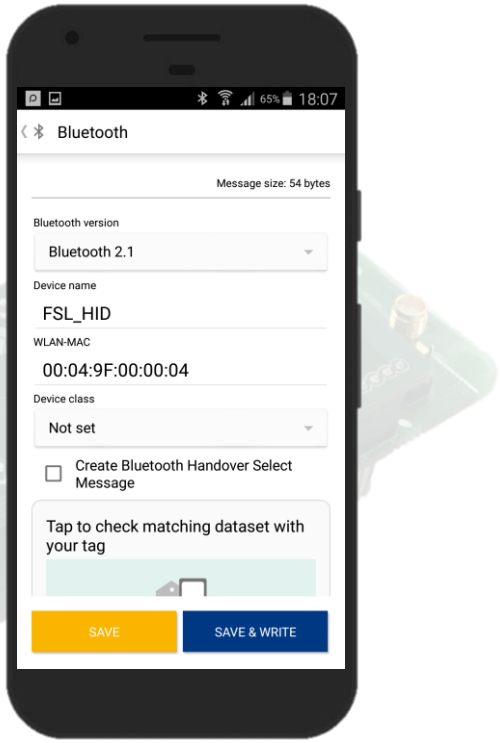
How to alternatively configure NTAG21x / NTAG I²C *plus* for Bluetooth pairing

NFC TagWriter by NXP









Download NXP TagWriter application from Play Store

Go to the “Write tags -> New Dataset ->

Go to -> Bluetooth

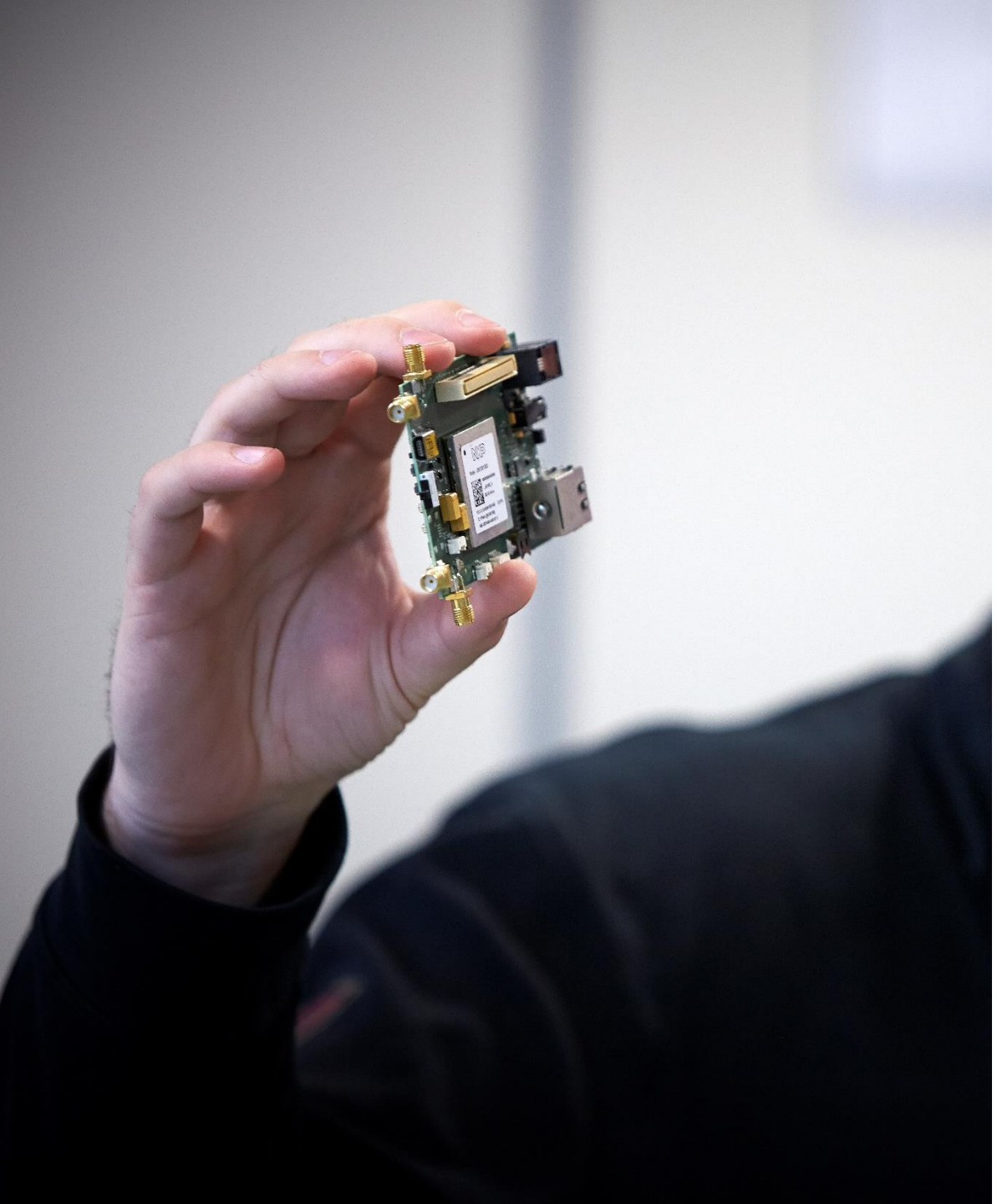
Fill the Device name with:

- **“FSL_HID”**
- **MAC is “00:04:9F:00:00:04”**

And tap the NTAG board

Training Mobile Knowledge

Wrap up and Q&A



Summary of available resources

- BLE pairing with NFC on KW41 and NTAG I²C *plus* source code
  www.nxp.com/downloads/en/snippets-boot-code-headers-monitors/SW4223.zip
- NTAG I²C *plus* kit for Arduino pinout
  www.nxp.com/demoboard/OM23221ARD
- FRDM-KW41Z board
  www.nxp.com/demoboard/FRDM-KW41Z
- NXP NFC community
  https://community.nxp.com/community/nfc

MobileKnowledge

Software development in Android and iOS
Embedded software for MCUs
JCOP, Java Card operating Systems
Hardware design and development
Digital, analog, sensor acquisition, power management
Wireless communications WiFi, ZigBee, Bluetooth, BLE
Contactless antenna RF design, evaluation and testing

MIFARE applications
End-to-end systems, readers and card-related designs
EMVco applications
Readers, cards, design for test compliancy (including PCI)
Secure Element management
GlobalPlatform compliant backend solutions
Secure services provisioning OTA, TSM services

We help companies leverage the mobile and **contactless revolution**

MobileKnowledge

Roc Boronat 117, P3M3
08018 Barcelona
(Spain)

Get in touch with us
www.themobileknowledge.com
mk@themobileknowledge.com