NFC FOR CONSUMABLES AND ACCESSORIES

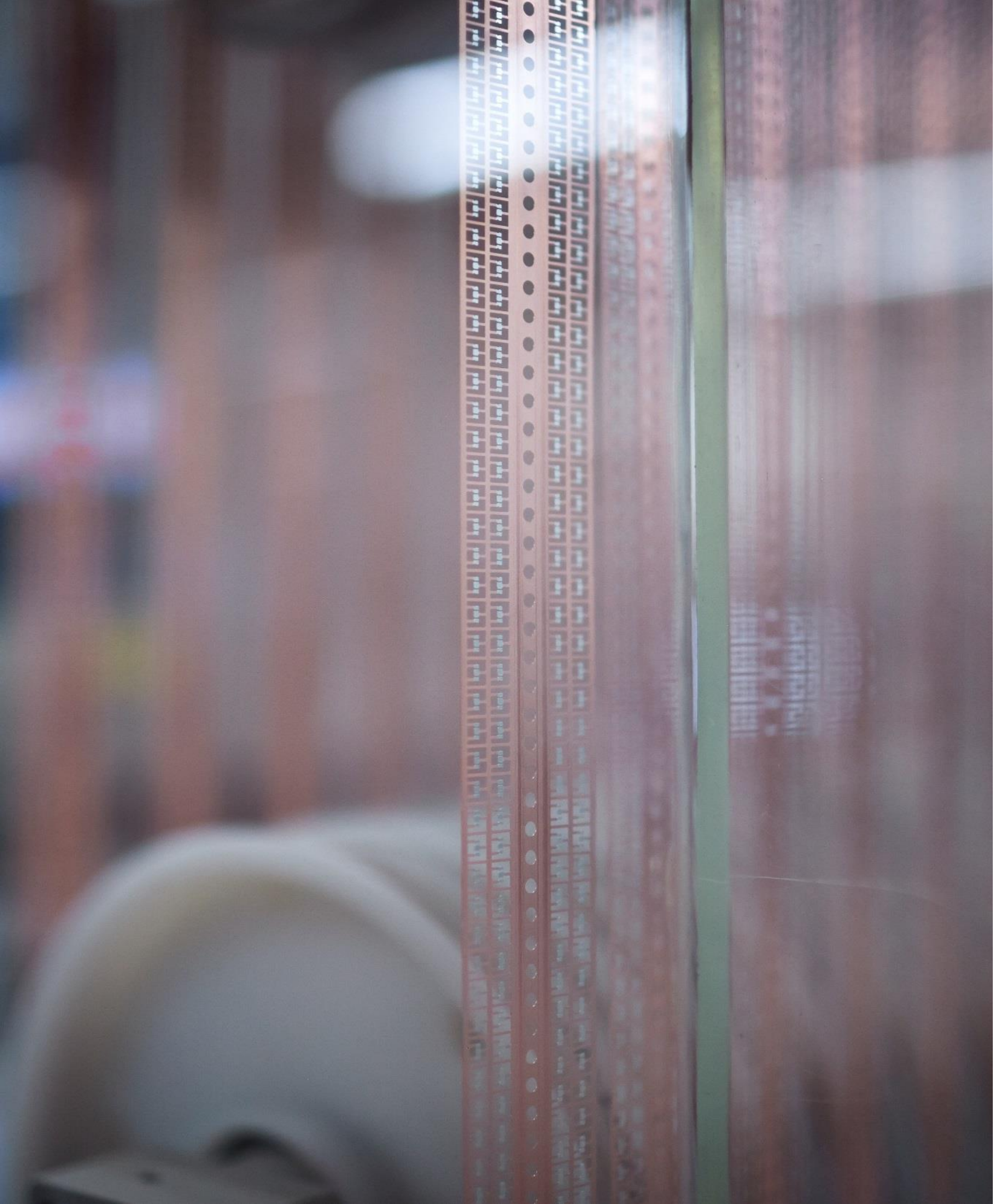
WEBINAR SERIES:

HOW TO DEVELOP NFC APPLICATIONS

JORDI JOFRE
NFC READERS
NFC EVERYWHERE
22/02/2018

PUBLIC

SECURE CONNECTIONS
FOR A SMARTER WORLD



Agenda

- NFC for product authentication & identification
- NFC portfolio for product authentication & identification
- NFC Nutshell Kit
- Consumable authentication sample application logic

NFC for product authentication & identification

NFC for product authentication & identification

Combat counterfeits by authenticating accessories

Create more interactive and personal experiences

Order branded replacements/ consumables with a single tap

Automatically choose the right tool every time

NFC Benefits

- Adjust settings of the main unit based on the accessory attached
- Ensure authenticity of the consumable / fight counterfeits
- Improve accuracy by storing calibration data on the tag
- Identify users and immediately provide personalized settings
- Send notifications when accessories are nearing replacement

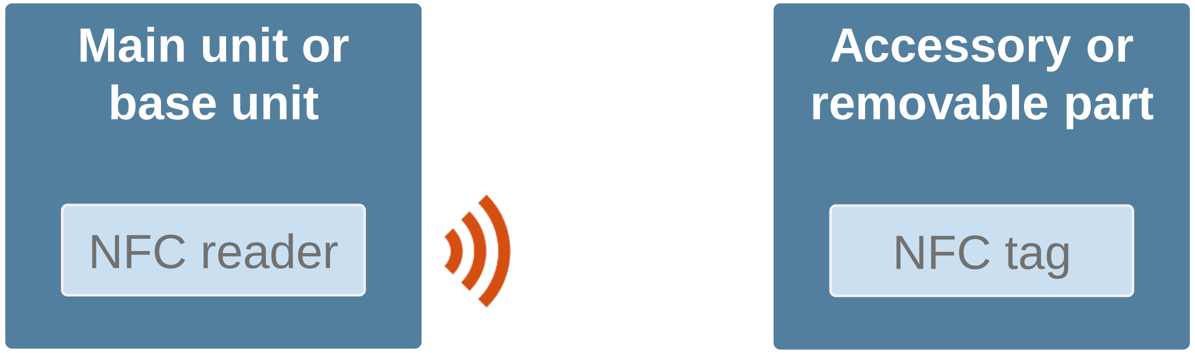
How NFC works in product authentication & identification

Use case

- Validate the originality of the consumable / accessory
- Optionally, configure the device with related settings

Goals

- Ensure originality including recurring revenues on consumables
- Enhance consumer experience and convenience
- Ensure product safety



Data read by NFC Reader inside the base unit, e.g. fridge, blender, then sent to MCU

NFC Tag in the removable part, e.g. brush head, water or air filter, …

NFC success stories

High-end blender

NFC reader: in the base unit

NFC tag: in the jug/container

Application: Check lid is closed before starting configuration: settings

Face brush

NFC reader: in the handle

NFC tag: in the brush heads

Application: Automatically configure the brush speed & spinning parameters

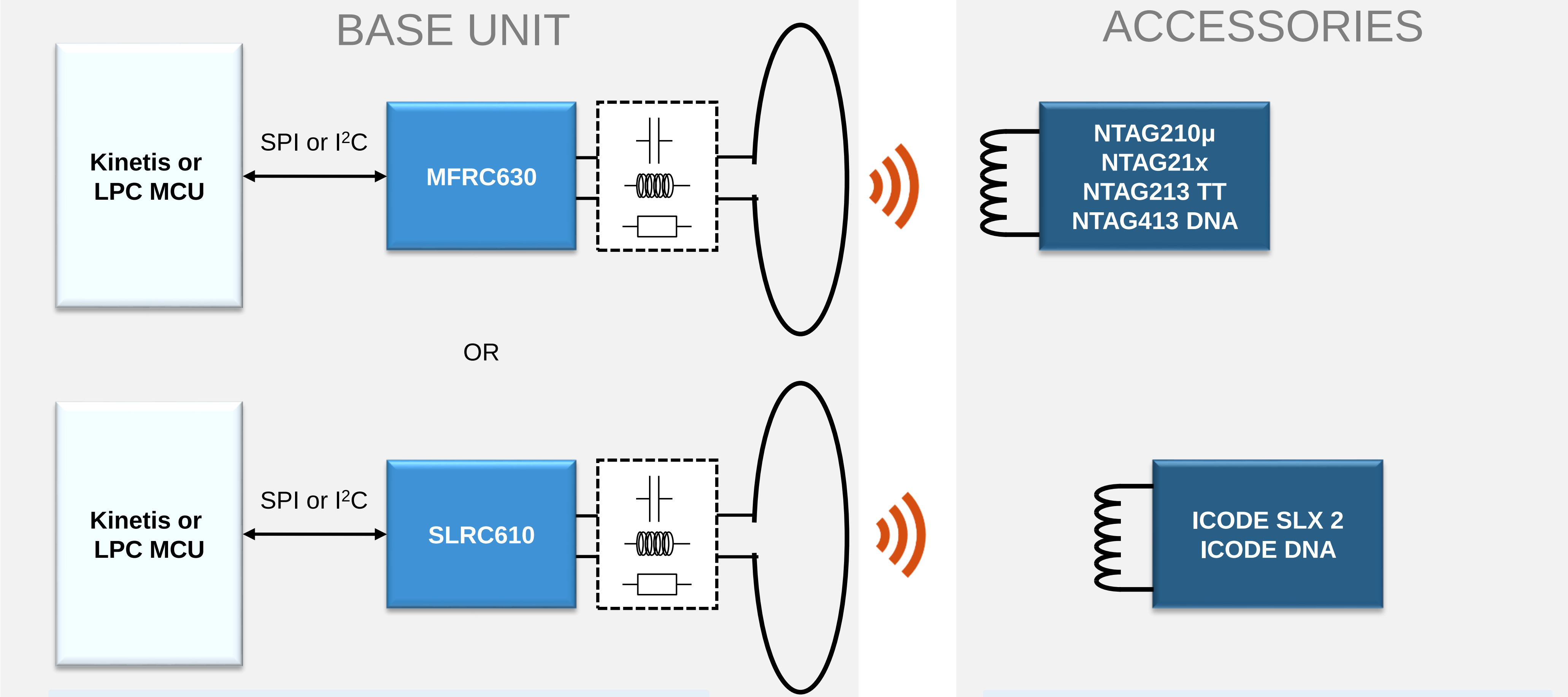
Water filter for fridges

NFC reader: in the fridge base

NFC tag: in the water filter

Application: Check that the right & genuine water filter is in place

How to implement the use case



BASE UNIT

Kinetis or LPC MCU

SPI or I^{2}C

MFRC630

OR

Kinetis or LPC MCU

SPI or I^{2}C

SLRC610

ACCESSORIES

NTAG210µ
NTAG21x
NTAG213 TT
NTAG413 DNA

ICODE SLX 2
ICODE DNA

Great NFC performance, low power consumption readers. Highly flexible frontend for all solutions.

Tags provide a great way to identify and authenticate consumables & accessories.

Solution selection guidelines

What do you need to achieve?

- Brand protection (only original supplies work), automation, safety / security issues, etc.

Which are your security needs?

- Identification, authentication, signature, integrity check, encrypted communication, etc.

What reading distance do you need?

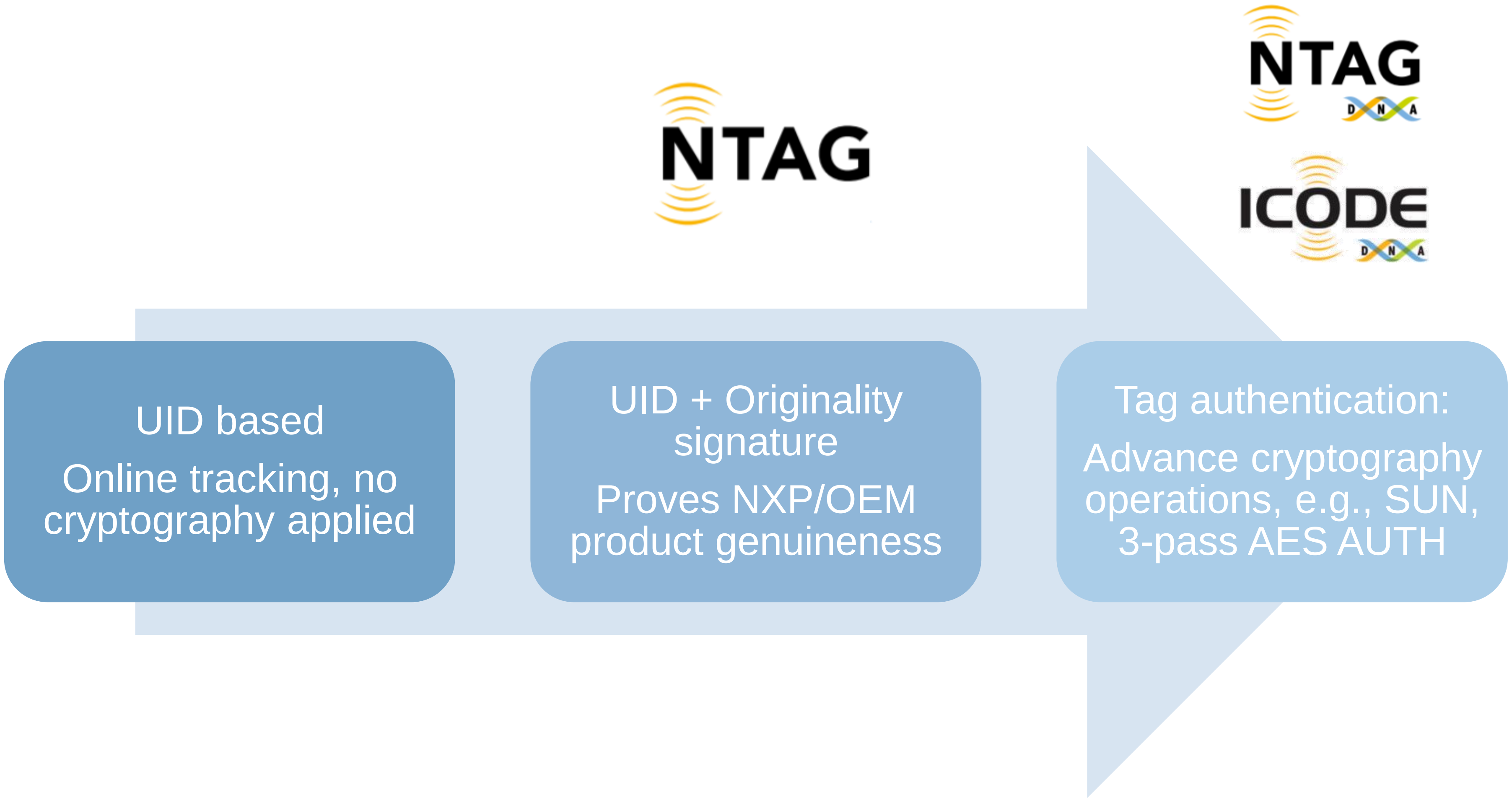
- A few cm, a few tenths of cm, etc.

Do you have space constraints in the product?

- Directly impacts the antenna size.

NXP portfolio for product authentication & identification

NFC security features in NXP portfolio



UID based

Online tracking, no cryptography applied

UID + Originality signature

Proves NXP/OEM product genuineness

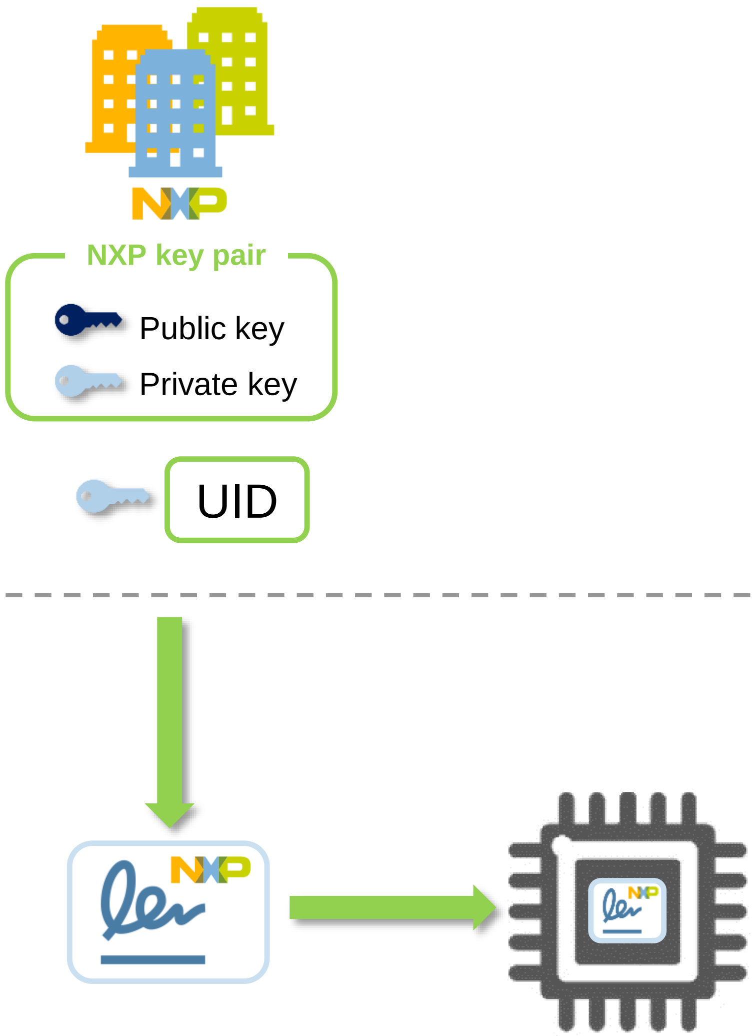
Tag authentication:

Advance cryptography operations, e.g., SUN, 3-pass AES AUTH

Originality Signature generation during IC production

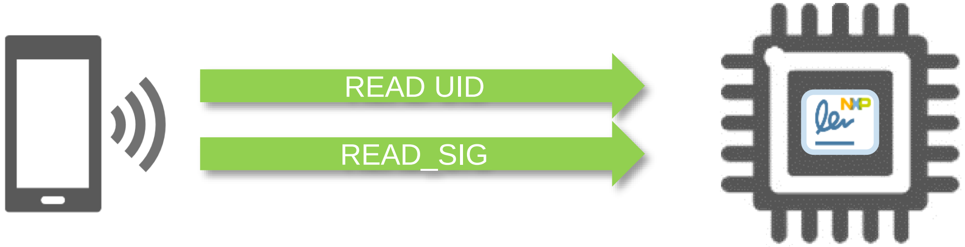
1 Unique ID per IC is signed by NXP

2 Signature is stored inside the IC

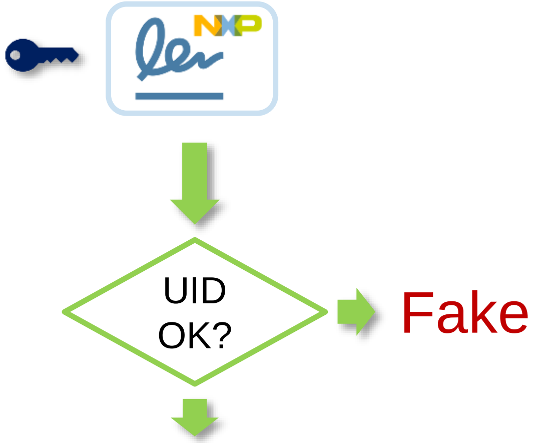


Originality Signature verification

1 UID and signature are retrieved



2 Signature is verified with the IC UID



Fake

Original device

NXP key pair

Public key

Private key

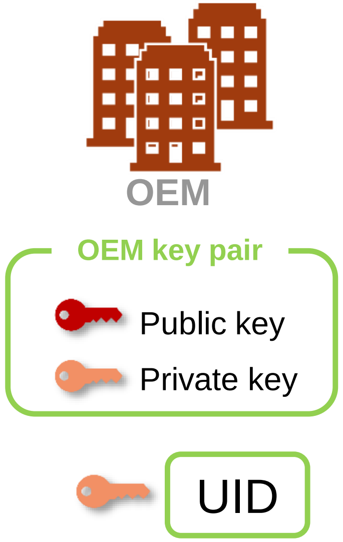
OEM Customizable originality signature

NTAG (*)

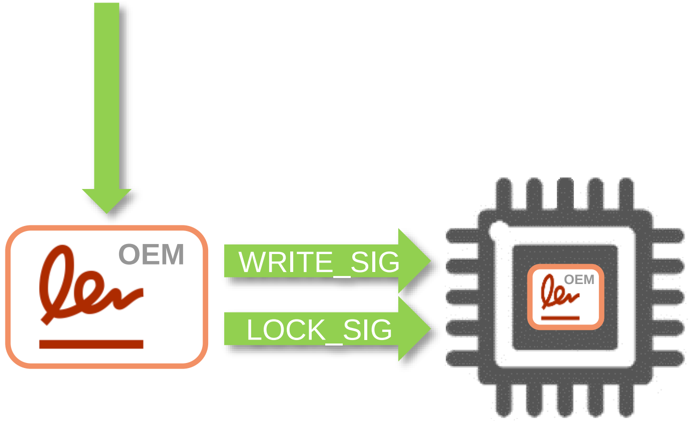
ICODE DNA

(*) Only some NTAG family members support this feature

1 Unique ID per IC is signed by OEM

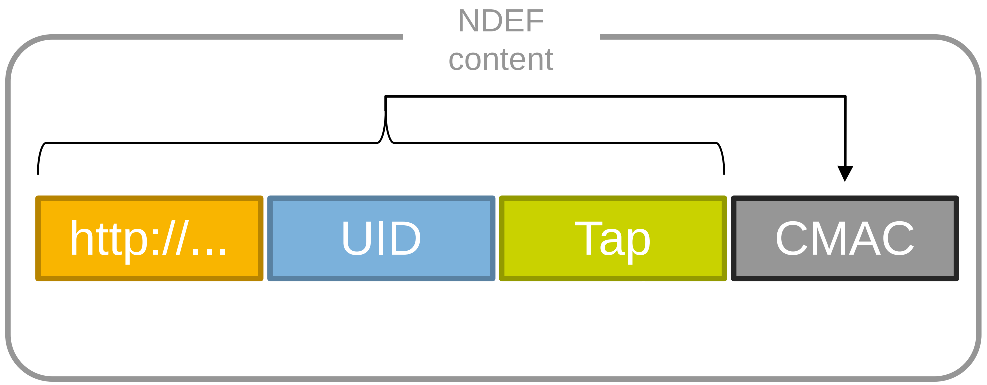


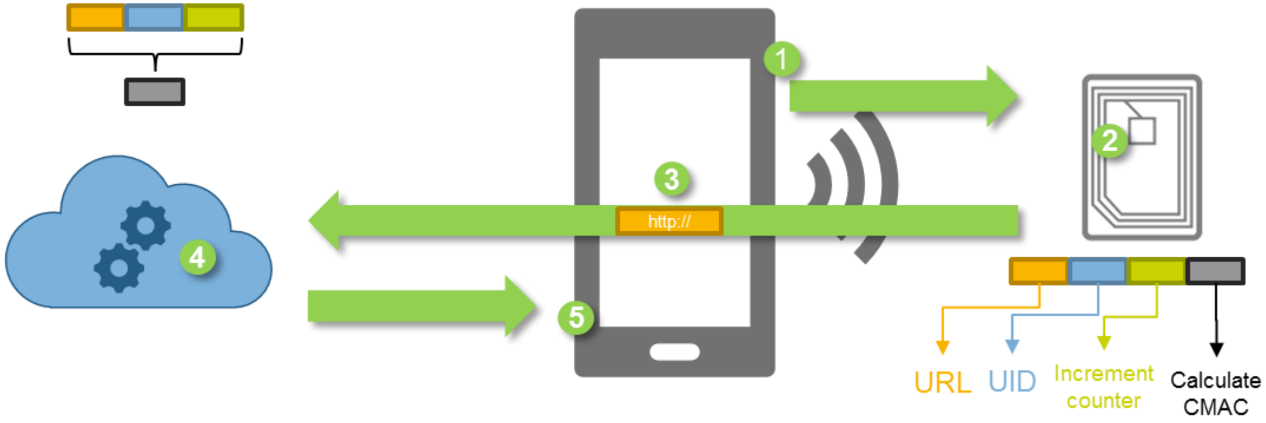
2 Signature is stored and locked inside the IC



Secure Unique NFC Message (SUN)

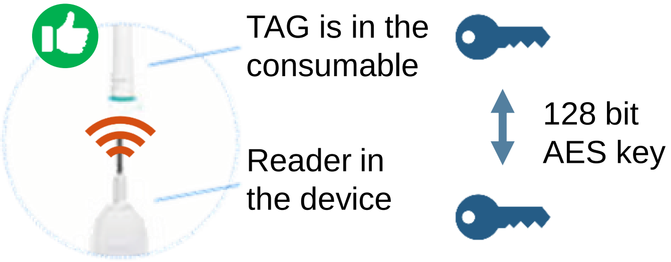
- Unique NDEF message generated each tap
- Incremental NFC counter each tap available
- Direct connection to web-service with no app required
- AES based dynamic CMAC as part of the NDEF data





AES 3-pass mutual authentication

- Tag and reader authentication
- 3 AES 128-bit application keys available
- Used key is known to both receiver and sender



TAG is in the consumable

Reader in the device

128 bit AES key

Advanced Encryption Standard is a specification for symmetric encryption

ISO/IEC 29167 defines procedures for tag authentication using AES-128

AES 3-pass mutual authentication

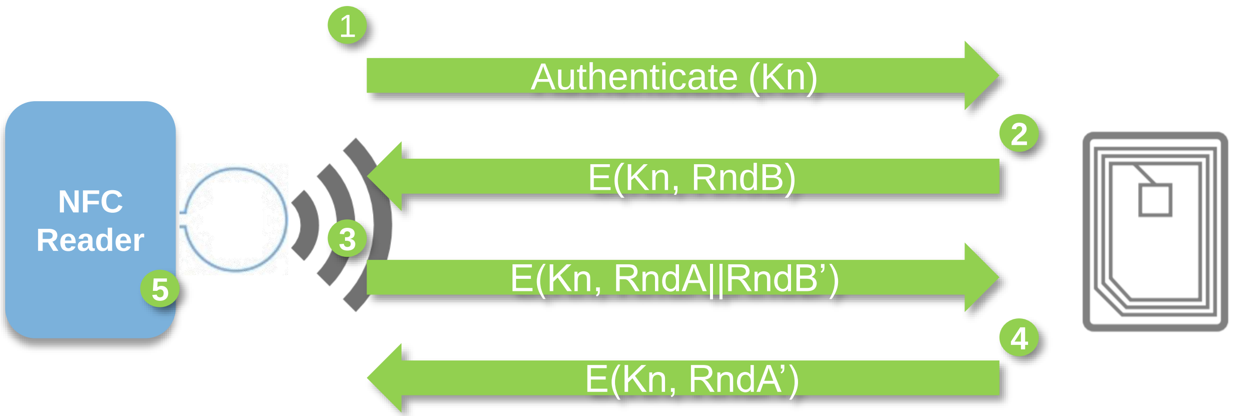
❶ Reader sends authentication command with key number to use

❷ Tag generates random challenge, encrypts it and sends the response

❸ Reader decrypts the challenge, combines it with a new challenge, encrypts the result and sends the response

❹ Tag decrypts the message, and sends the reader's challenge encrypted

❺ If all challenges have been successful, both ends are now authenticated and have a shared secret



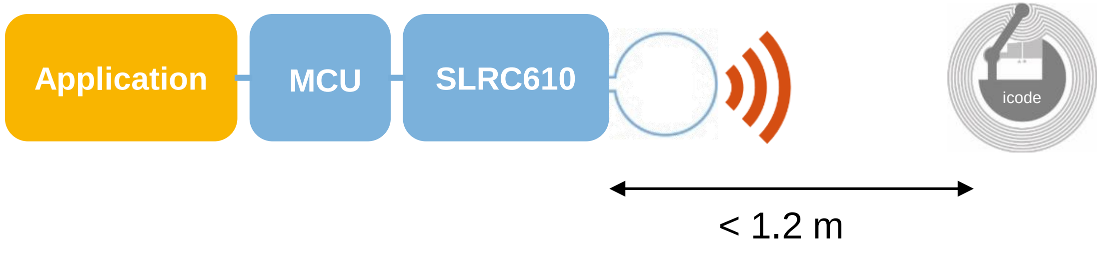
Tag Comparison

	NTAG21x	NTAG210µ	NTAG413 DNA	ICODE DNA
Type	Type 2	Type 2	Type 4	Type 5
Operating distance up to	10 cm	10 cm	10 cm	1.2 m
Originality signature	32 Bytes (NXP signature)	32 Bytes customizable	56 Bytes (NXP signature)	32 Bytes customizable
3-pass AES Auth	✗	✗	✓	✓
SUN	✗	✗	✓	✗
Memory	144-888 Bytes	64 Bytes	128 Bytes	256 Bytes

NFC frontends

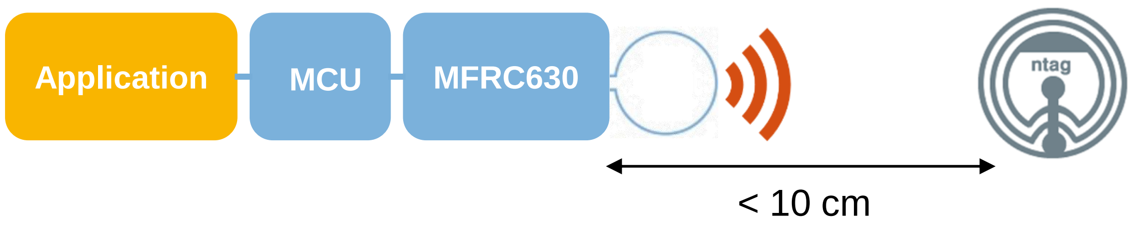
SLRC610 plus

- Multiprotocol: ISO/IEC 15693, ISO/IEC 18000-3
- Supporting vicinity products **ICODE**
- Host interfaces: SPI I²C, UART
- Separate I²C bus for a SAM



MFRC630 plus

- RF standard compliance: ISO 14443A
- Recommended solution for **NTAG** and **MIFARE**
- Host interfaces: SPI I²C, UART
- Separate I²C bus for a SAM



NFC Nutshell Kit

NFC Nutshell Kit introduction

Need to add NFC into your products ?

The NFC Nutshell Kit modules are specifically designed for:

- NFC technology integration / retrofitting into existing or new product designs
- Building up of NFC enabled demonstrators
- NFC technology evaluation
- Application testing and development

Developed by GMMC, the kit contains several modules covering most of NXP portfolio for NFC solutions. GMMC is an approved engineering consultant of NXP for NFC (https://nxp.surl.ms/NFC_AEC)

GMMC we deliver for your success

Benefits & features

- Nano sized modules for space constrained environments
- Flexible configuration to adapt to different conditions
- Support of most popular NXP NFC reader/writer ICs
  - CLRC663plus family, PN5180, PN7150, PN7462 family
- Compatibility with existing NXP NFC and MCU development tools
  - NFC Cockpit, RFIDDiscover, MCU Expresso, LPC Link2

Modules

Host interfaces:

- USB Plug
- Programmable converter USB to UART, I2C, SPI
- Signal Debug Extender

Microcontrollers:

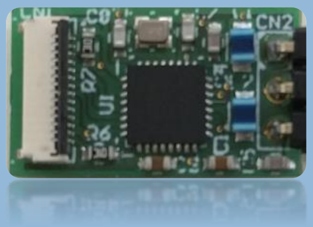
- LPC11u68 (JBD48)
- LPC1769

RF-Frontends:

- CLRC663 plus family, including SLRC610, MFRC630 MFRC631
- PN5180

RF-Frontend with integrated MCU:

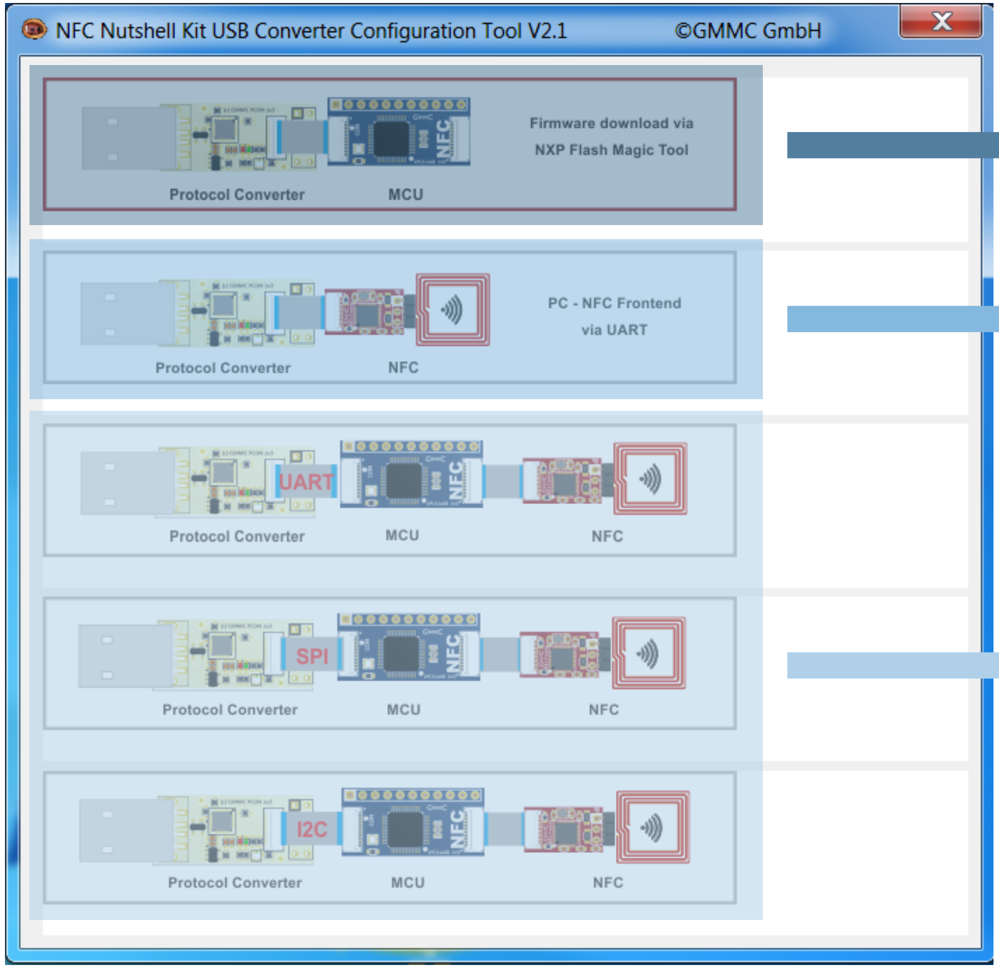
- PN7462 (Q2 2018)
- PN7150 (Q2 2018)



Antennas:

- 20x10 mm
- 20x20 mm
- 40x40 mm
- 72x48 mm
- Twisted wire connection between antenna and RF modules

Modes of operation for USB protocol converter



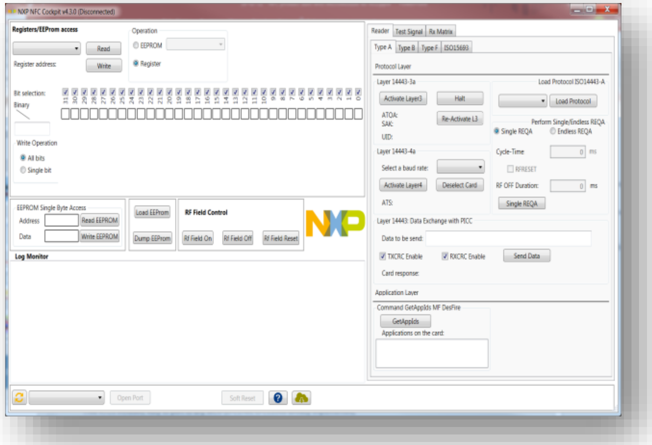
MCU stays in programming mode, only for MCU flashing

No MCU is used. The computer talks with the frontend via UART

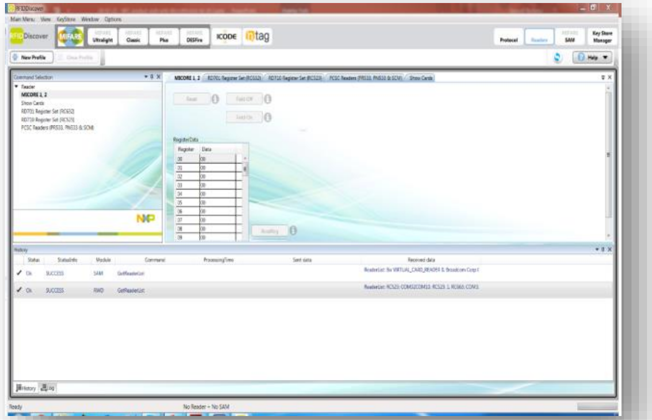
The computer can talk with the MCU over the specified protocol

Supported NXP development tools

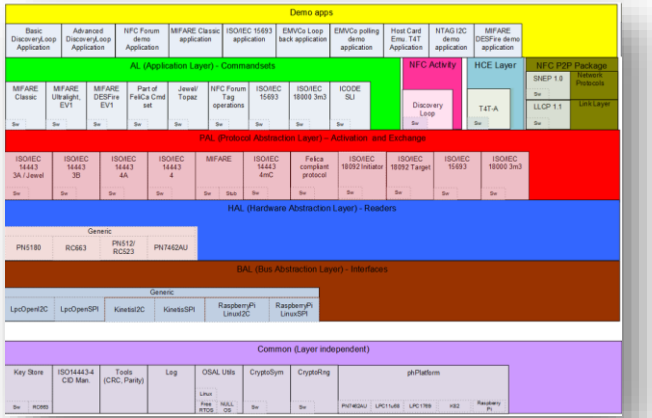
NFC Cockpit



RFIDDiscover



NFC Reader library



More information on NFC-Cockpit

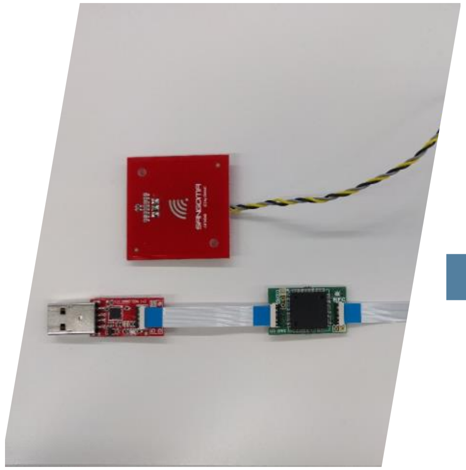
More information on RFIDDiscover

More information on NFC Reader library

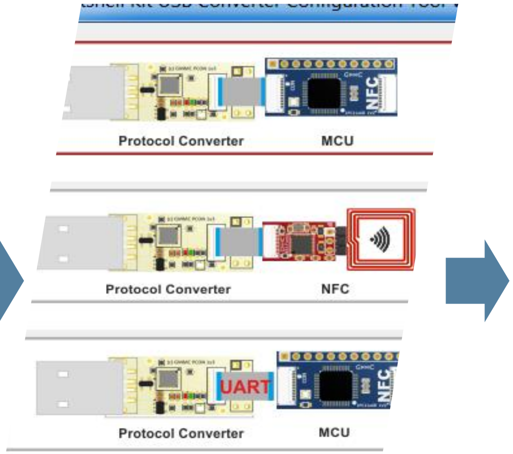
Consumable authentication

Sample application logic

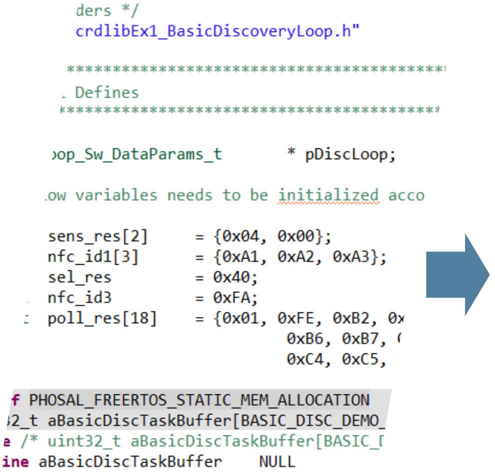
Running the NFC reader library in the Nutshell Kit



1. Prepare the hardware

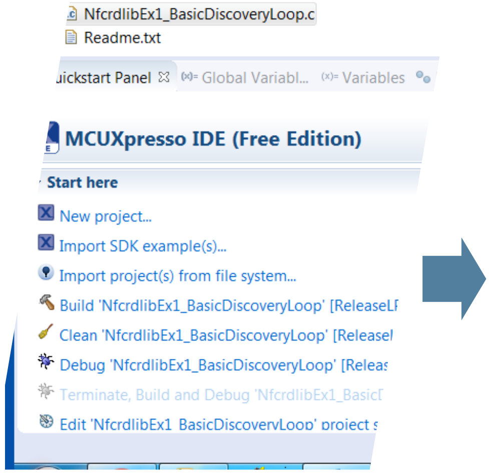


2. Configure USB converter



```
ders */
crdlibEx1_BasicDiscoveryLoop.h"

****************************************
. Defines
****************************************

oop_Sw_DataParams_t       * pDiscLoop;

.ow variables needs to be initialized acco

 sens_res[2]      = {0x04, 0x00};
 nfc_id1[3]       = {0xA1, 0xA2, 0xA3};
 sel_res          = 0x40;
 nfc_id3          = 0xFA;
 poll_res[18]     = {0x01, 0xFE, 0xB2, 0x
                          0xB6, 0xB7, (
                          0xC4, 0xC5,

f PHOSAL_FREERTOS_STATIC_MEM_ALLOCATION
2_t aBasicDiscTaskBuffer[BASIC_DISC_DEMO_
e /* uint32_t aBasicDiscTaskBuffer[BASIC_[
ine aBasicDiscTaskBuffer    NULL
```

3. Development



NfcrdlibEx1_BasicDiscoveryLoop.c
Readme.txt

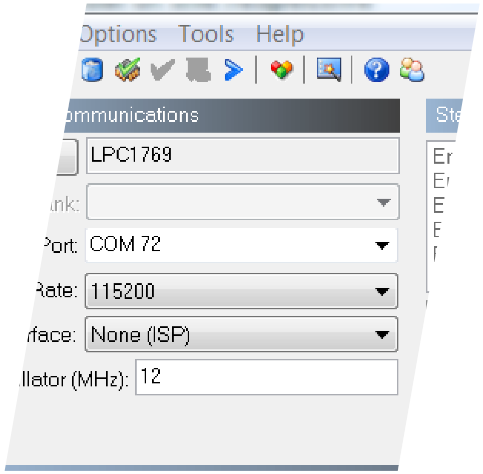
uickstart Panel ☒ (x)= Global Variabl... (x)= Variables

MCUXpresso IDE (Free Edition)

Start here

- New project...
- Import SDK example(s)...
- Import project(s) from file system...
- Build 'NfcrdlibEx1_BasicDiscoveryLoop' [ReleaseLF
- Clean 'NfcrdlibEx1_BasicDiscoveryLoop' [Releasel
- Debug 'NfcrdlibEx1_BasicDiscoveryLoop' [Releas
- Terminate, Build and Debug 'NfcrdlibEx1_BasicI
- Edit 'NfcrdlibEx1_BasicDiscoveryLoop' project s

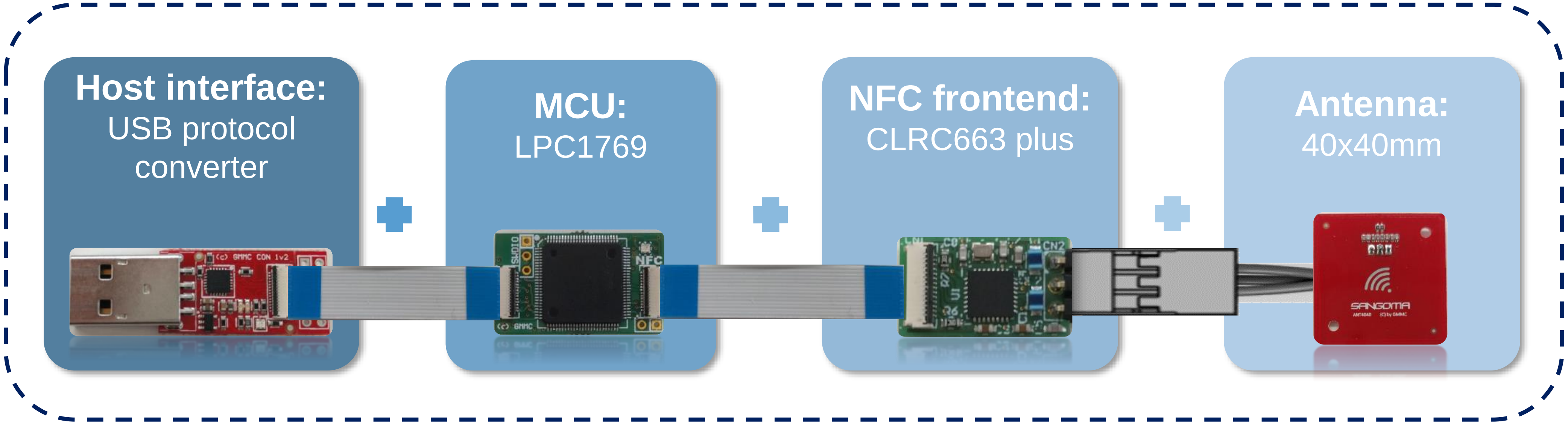
4. Build project in MCUXpresso



LPC1769
ank:
Port: COM 72
Rate: 115200
rface: None (ISP)
llator (MHz): 12

5. Flash the MCU image

1. Prepare the hardware



Host interface: USB protocol converter

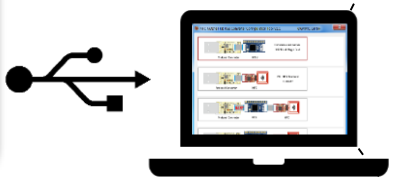
MCU: LPC1769

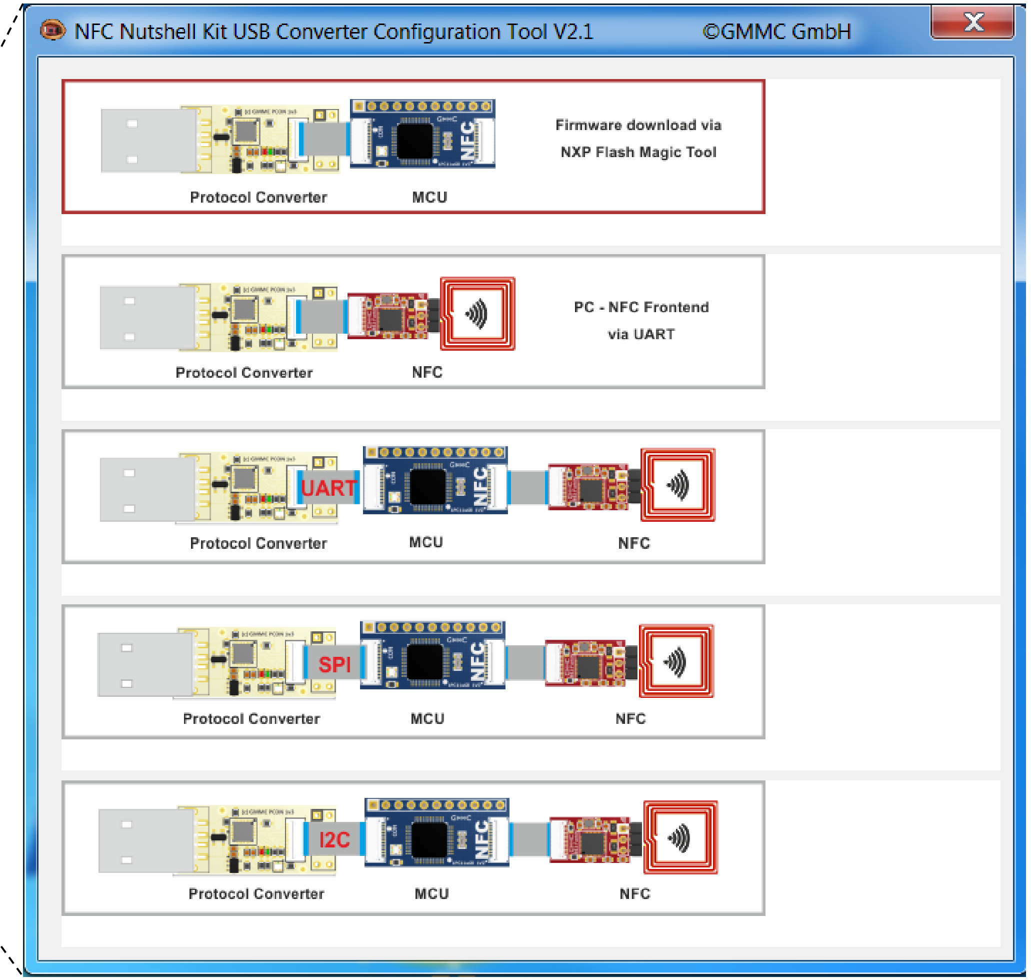
NFC frontend: CLRC663 plus

Antenna: 40x40mm

2. Configure USB converter

Host interface:
USB protocol
converter

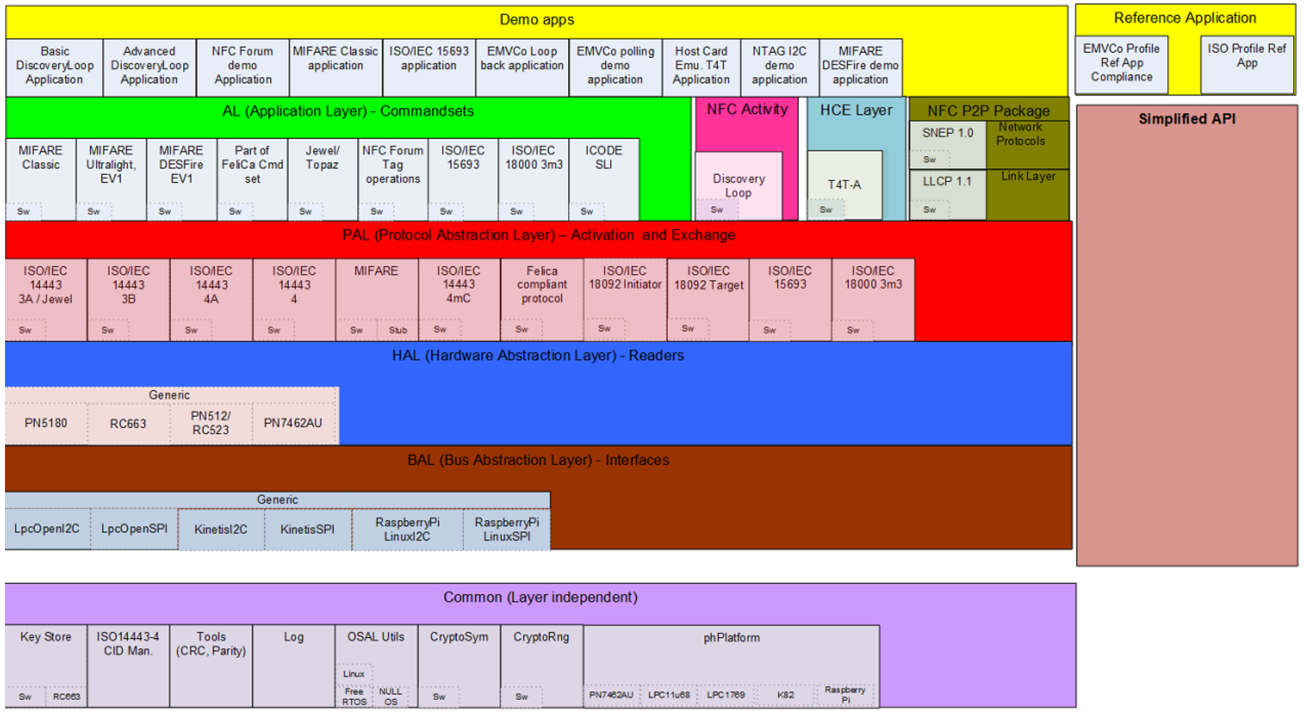




NFC Nutshell Kit USB Converter Configuration Tool V2.1 ©GMMC GmbH

3. Development: NFC Reader Library

NFC Reader Library

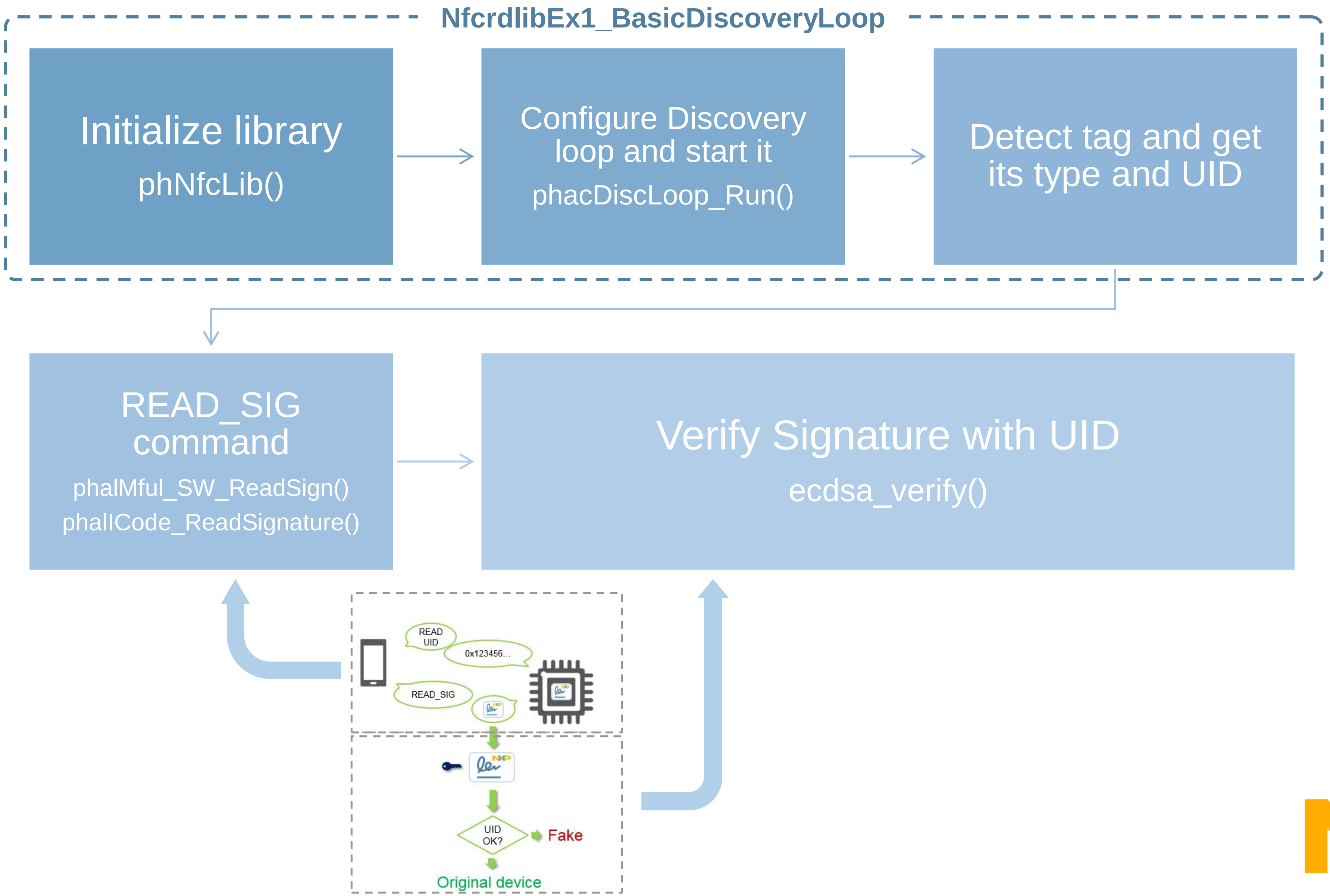


Software examples

Example 1: BasicDiscoveryLoop
Example 2: AdvancedDiscoveryLoop
Example 3: NFCForum
Example 4: MIFARE Classic
Example 5: ISO15693
Example 6: EMVCo Loopback
Example 7: EMVCo Polling
Example 8: HCE T4T
Example 9: NTAG I2C
Example 10: SimplifiedAPI_EMVCo
Example 11: SimplifiedAPI_ISO

The NFC Reader Library is everything you need to create your own software stack and application for a contactless reader

3. Development: Originality signature verification



NfcrdlibEx1_BasicDiscoveryLoop

Initialize library

phNfcLib()

Configure Discovery loop and start it

phacDiscLoop_Run()

Detect tag and get its type and UID

READ_SIG command

phalMful_SW_ReadSign()

phalICode_ReadSignature()

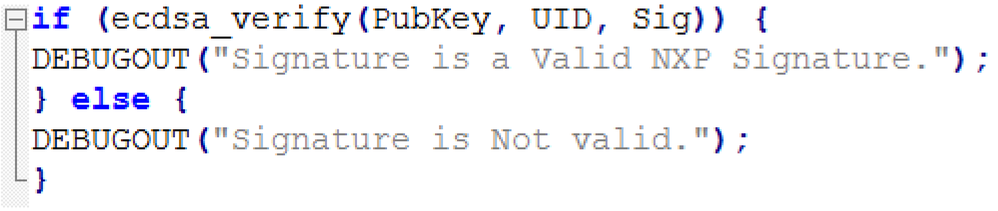
Verify Signature with UID

ecdsa_verify()

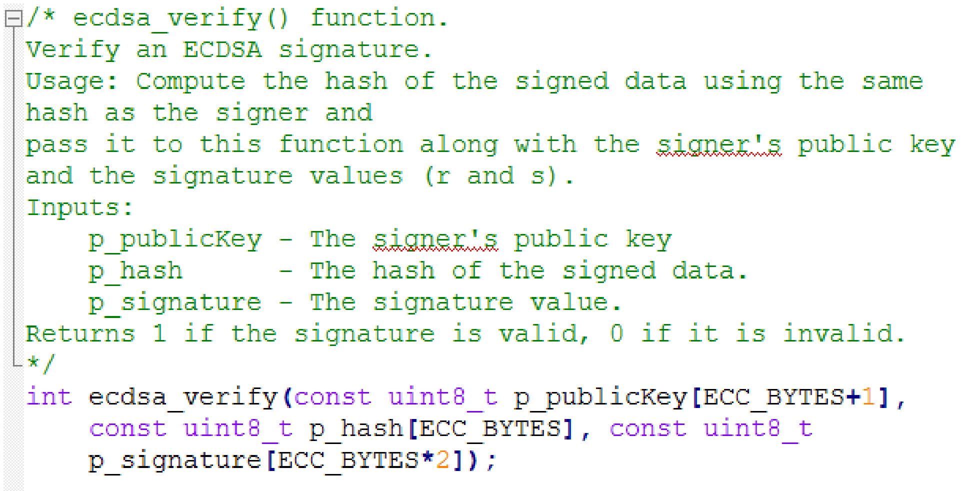
3. Development: Signature verification

- Reader library does not include asymmetric crypto
- Easy-ecc: a simple and secure ECDH and ECDSA library written in C
- Easy integration and use

https://github.com/esxgx/easy-ecc



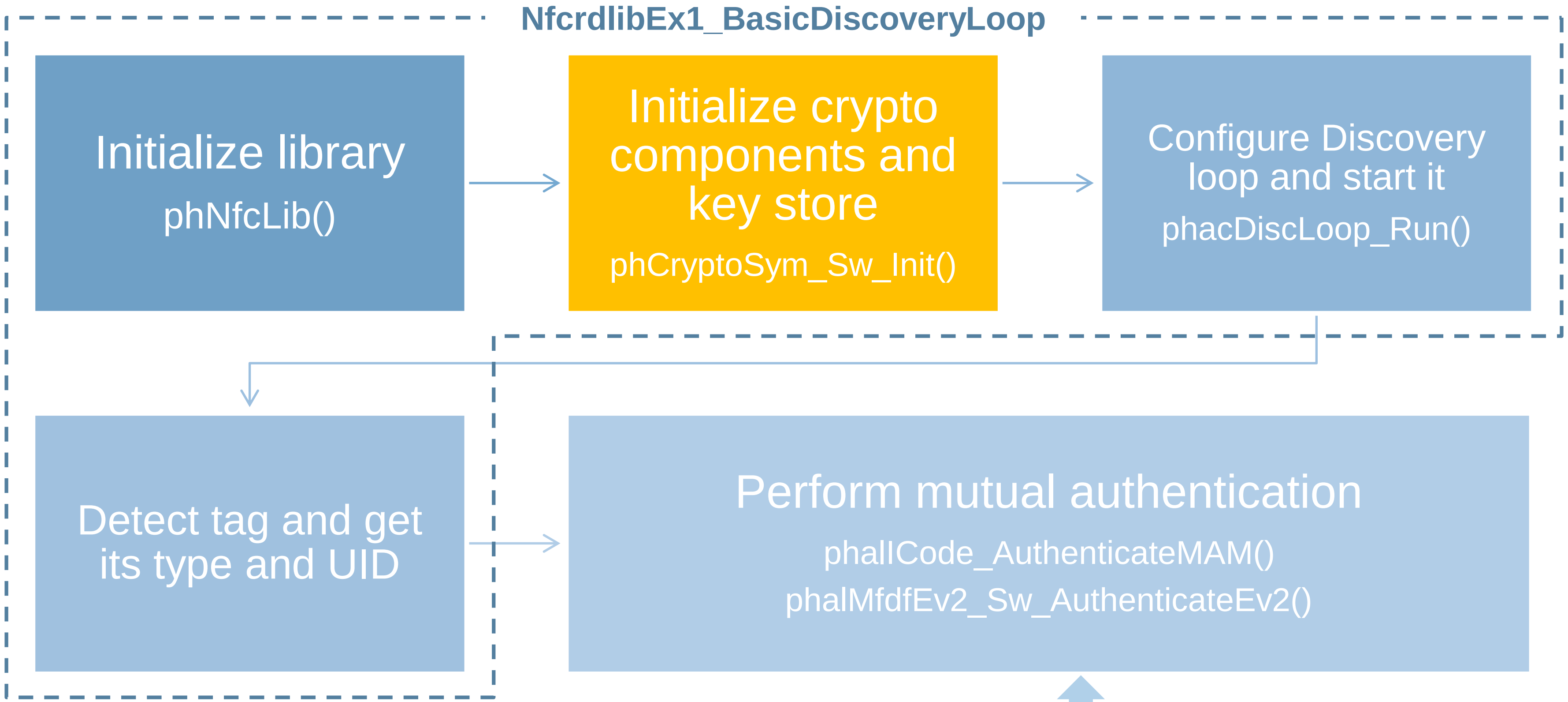
```
if (ecdsa_verify(PubKey, UID, Sig)) {
DEBUGOUT("Signature is a Valid NXP Signature.");
} else {
DEBUGOUT("Signature is Not valid.");
}
```



```
/* ecdsa_verify() function.
Verify an ECDSA signature.
Usage: Compute the hash of the signed data using the same
hash as the signer and
pass it to this function along with the signer's public key
and the signature values (r and s).
Inputs:
    p_publicKey - The signer's public key
    p_hash      - The hash of the signed data.
    p_signature - The signature value.
Returns 1 if the signature is valid, 0 if it is invalid.
*/
int ecdsa_verify(const uint8_t p_publicKey[ECC_BYTES+1],
    const uint8_t p_hash[ECC_BYTES], const uint8_t
    p_signature[ECC_BYTES*2]);
```

More information can be found in AN11350 NTAG Originality Signature Validation (1.2) document

3. Development: 3-pass mutual authentication



NfcrdlibEx1_BasicDiscoveryLoop

Initialize library

phNfcLib()

Initialize crypto components and key store

phCryptoSym_Sw_Init()

Configure Discovery loop and start it

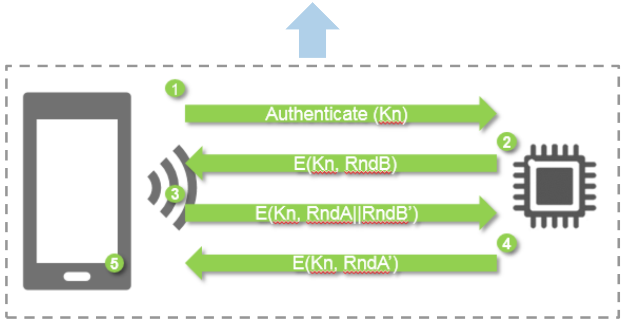
phacDiscLoop_Run()

Detect tag and get its type and UID

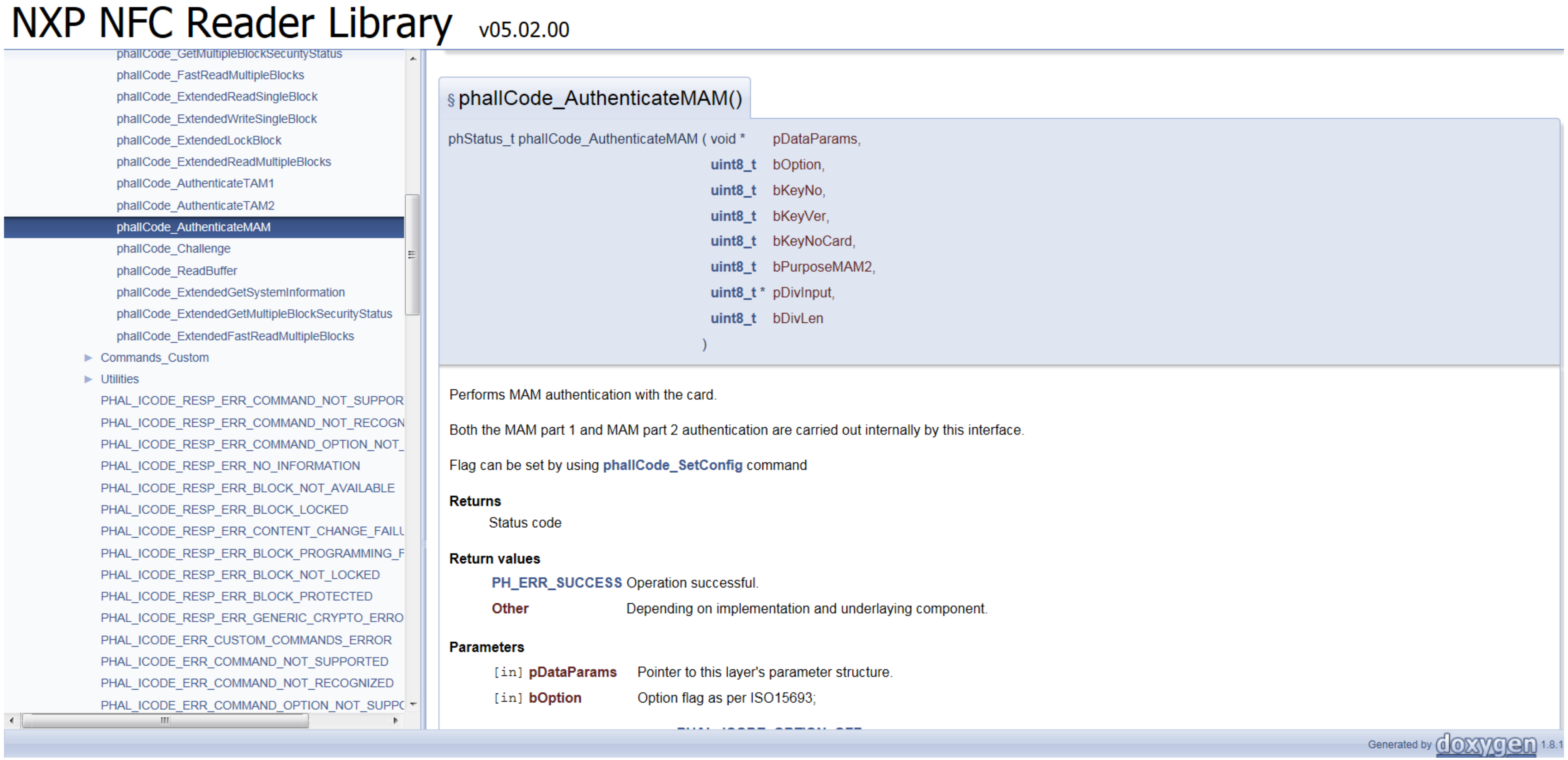
Perform mutual authentication

phalICode_AuthenticateMAM()

phalMfdfEv2_Sw_AuthenticateEv2()



3. Development: 3-pass mutual authentication



NXP NFC Reader Library v05.02.00

phalICode_GetMultipleBlockSecurityStatus
phalICode_FastReadMultipleBlocks
phalICode_ExtendedReadSingleBlock
phalICode_ExtendedWriteSingleBlock
phalICode_ExtendedLockBlock
phalICode_ExtendedReadMultipleBlocks
phalICode_AuthenticateTAM1
phalICode_AuthenticateTAM2
phalICode_AuthenticateMAM
phalICode_Challenge
phalICode_ReadBuffer
phalICode_ExtendedGetSystemInformation
phalICode_ExtendedGetMultipleBlockSecurityStatus
phalICode_ExtendedFastReadMultipleBlocks
▶ Commands_Custom
▶ Utilities
PHAL_ICODE_RESP_ERR_COMMAND_NOT_SUPPOR
PHAL_ICODE_RESP_ERR_COMMAND_NOT_RECOGN
PHAL_ICODE_RESP_ERR_COMMAND_OPTION_NOT_
PHAL_ICODE_RESP_ERR_NO_INFORMATION
PHAL_ICODE_RESP_ERR_BLOCK_NOT_AVAILABLE
PHAL_ICODE_RESP_ERR_BLOCK_LOCKED
PHAL_ICODE_RESP_ERR_CONTENT_CHANGE_FAILU
PHAL_ICODE_RESP_ERR_BLOCK_PROGRAMMING_F
PHAL_ICODE_RESP_ERR_BLOCK_NOT_LOCKED
PHAL_ICODE_RESP_ERR_BLOCK_PROTECTED
PHAL_ICODE_RESP_ERR_GENERIC_CRYPTO_ERRO
PHAL_ICODE_ERR_CUSTOM_COMMANDS_ERROR
PHAL_ICODE_ERR_COMMAND_NOT_SUPPORTED
PHAL_ICODE_ERR_COMMAND_NOT_RECOGNIZED
PHAL_ICODE_ERR_COMMAND_OPTION_NOT_SUPPC

§ phalICode_AuthenticateMAM()

```
phStatus_t phalICode_AuthenticateMAM ( void *     pDataParams,
                                       uint8_t    bOption,
                                       uint8_t    bKeyNo,
                                       uint8_t    bKeyVer,
                                       uint8_t    bKeyNoCard,
                                       uint8_t    bPurposeMAM2,
                                       uint8_t *  pDivInput,
                                       uint8_t    bDivLen
                                     )
```

Performs MAM authentication with the card.

Both the MAM part 1 and MAM part 2 authentication are carried out internally by this interface.

Flag can be set by using **phalICode_SetConfig** command

Returns

Status code

Return values

PH_ERR_SUCCESS	Operation successful.
Other	Depending on implementation and underlaying component.

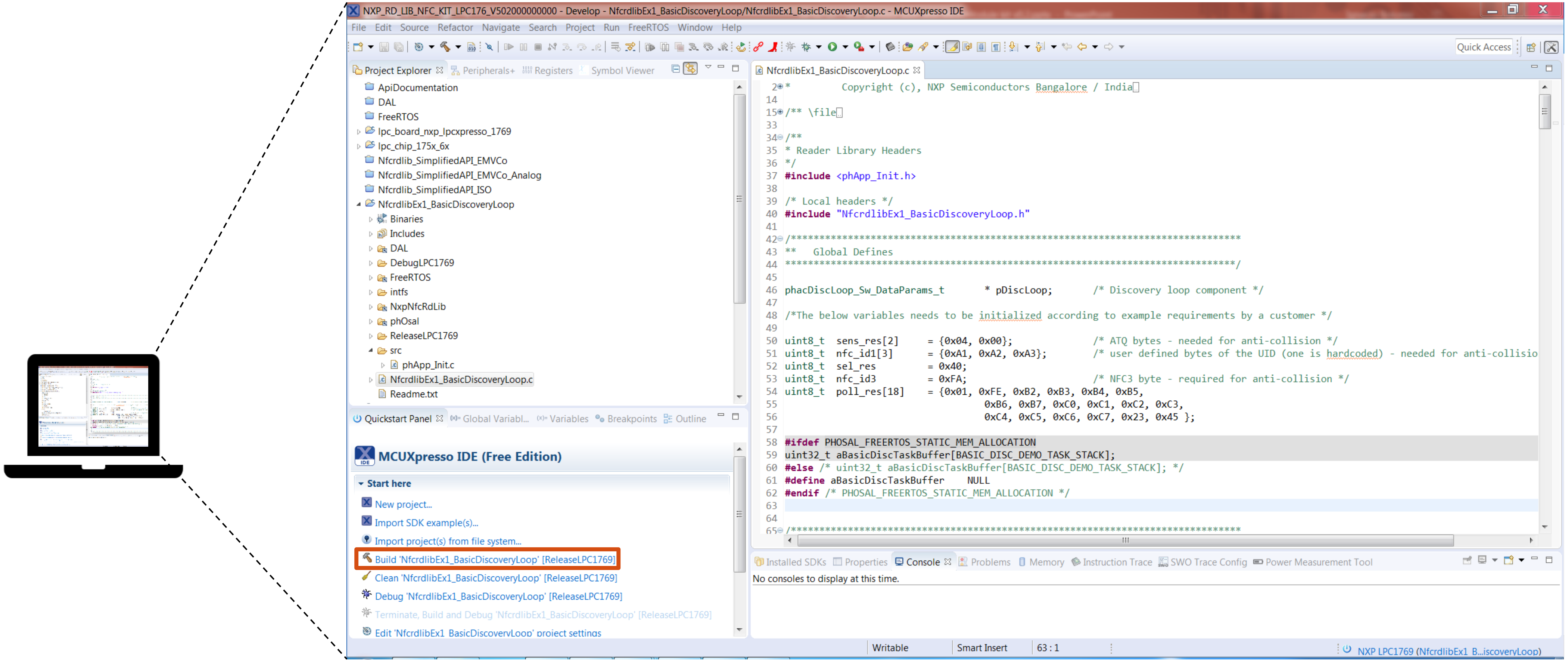
Parameters

[in]	**pDataParams**	Pointer to this layer's parameter structure.
[in]	**bOption**	Option flag as per ISO15693;

PHAL_ICODE_OPTION_OFF

Generated by doxygen 1.8.1

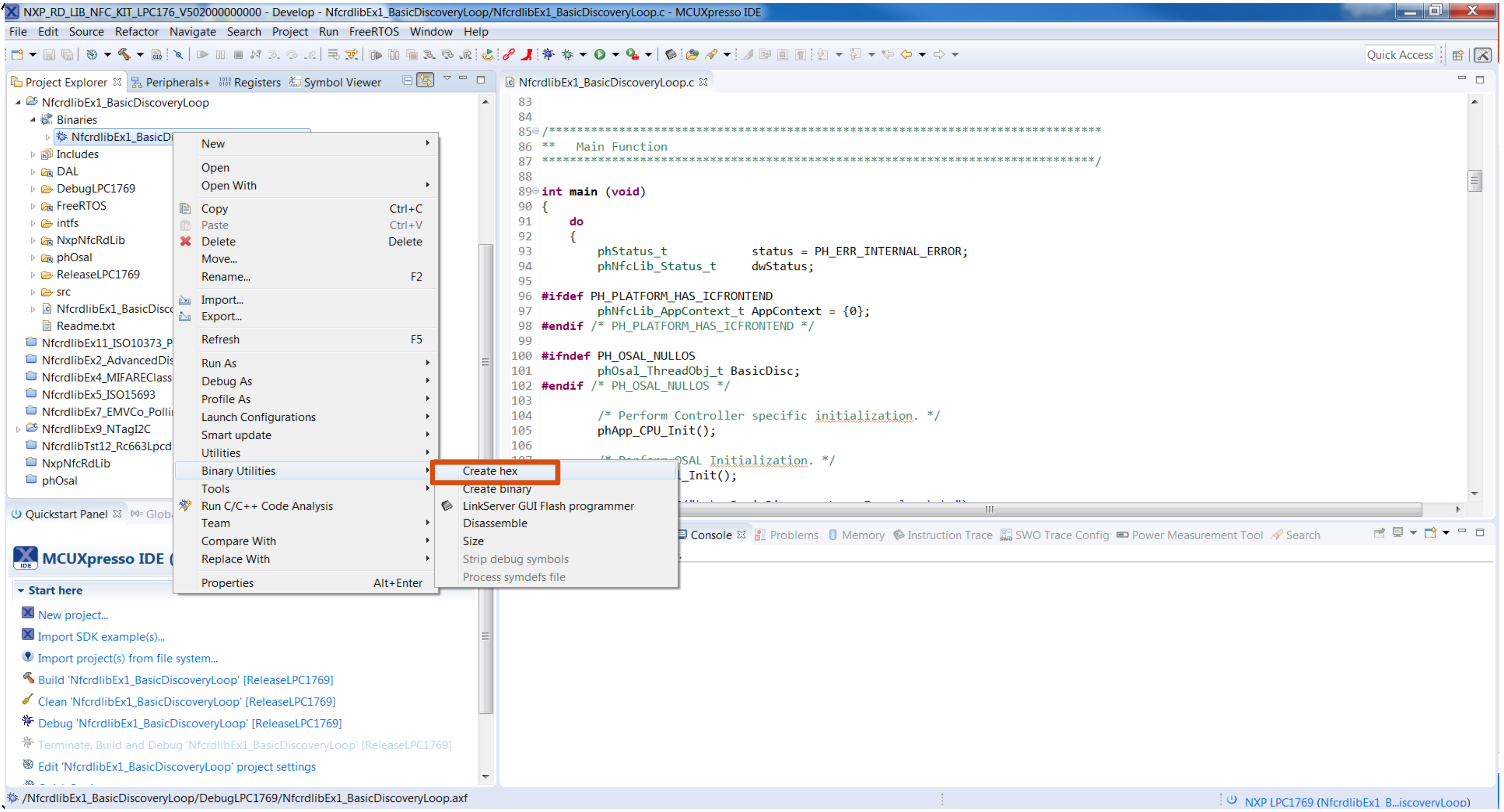
4. Build project in MCUXpresso


NXP_RD_LIB_NFC_KIT_LPC176_V502000000000 - Develop - NfcrdlibEx1_BasicDiscoveryLoop/NfcrdlibEx1_BasicDiscoveryLoop.c - MCUXpresso IDE
File Edit Source Refactor Navigate Search Project Run FreeRTOS Window Help
Quick Access
Project Explorer
Peripherals+
Registers
Symbol Viewer
ApiDocumentation
DAL
FreeRTOS
lpc_board_nxp_lpcxpresso_1769
lpc_chip_175x_6x
Nfcrdlib_SimplifiedAPI_EMVCo
Nfcrdlib_SimplifiedAPI_EMVCo_Analog
Nfcrdlib_SimplifiedAPI_ISO
NfcrdlibEx1_BasicDiscoveryLoop
Binaries
Includes
DAL
DebugLPC1769
FreeRTOS
intfs
NxpNfcRdLib
phOsal
ReleaseLPC1769
src
phApp_Init.c
NfcrdlibEx1_BasicDiscoveryLoop.c
Readme.txt
Quickstart Panel
Global Variabl...
Variables
Breakpoints
Outline
MCUXpresso IDE (Free Edition)
Start here
New project...
Import SDK example(s)...
Import project(s) from file system...
Build 'NfcrdlibEx1_BasicDiscoveryLoop' [ReleaseLPC1769]
Clean 'NfcrdlibEx1_BasicDiscoveryLoop' [ReleaseLPC1769]
Debug 'NfcrdlibEx1_BasicDiscoveryLoop' [ReleaseLPC1769]
Terminate, Build and Debug 'NfcrdlibEx1_BasicDiscoveryLoop' [ReleaseLPC1769]
Edit 'NfcrdlibEx1_BasicDiscoveryLoop' project settings
NfcrdlibEx1_BasicDiscoveryLoop.c
Installed SDKs
Properties
Console
Problems
Memory
Instruction Trace
SWO Trace Config
Power Measurement Tool
Writable
Smart Insert
63 : 1
NXP LPC1769 (NfcrdlibEx1_B...iscoveryLoop)

```c
*         Copyright (c), NXP Semiconductors Bangalore / India
/** \file

/**
* Reader Library Headers
*/
#include <phApp_Init.h>

/* Local headers */
#include "NfcrdlibEx1_BasicDiscoveryLoop.h"

/*******************************************************************************
**   Global Defines
*******************************************************************************/

phacDiscLoop_Sw_DataParams_t       * pDiscLoop;       /* Discovery loop component */

/*The below variables needs to be initialized according to example requirements by a customer */

uint8_t  sens_res[2]     = {0x04, 0x00};              /* ATQ bytes - needed for anti-collision */
uint8_t  nfc_id1[3]      = {0xA1, 0xA2, 0xA3};        /* user defined bytes of the UID (one is hardcoded) - needed for anti-collisio
uint8_t  sel_res         = 0x40;
uint8_t  nfc_id3         = 0xFA;                      /* NFC3 byte - required for anti-collision */
uint8_t  poll_res[18]    = {0x01, 0xFE, 0xB2, 0xB3, 0xB4, 0xB5,
                                   0xB6, 0xB7, 0xC0, 0xC1, 0xC2, 0xC3,
                                   0xC4, 0xC5, 0xC6, 0xC7, 0x23, 0x45 };

#ifdef PHOSAL_FREERTOS_STATIC_MEM_ALLOCATION
uint32_t aBasicDiscTaskBuffer[BASIC_DISC_DEMO_TASK_STACK];
#else /* uint32_t aBasicDiscTaskBuffer[BASIC_DISC_DEMO_TASK_STACK]; */
#define aBasicDiscTaskBuffer    NULL
#endif /* PHOSAL_FREERTOS_STATIC_MEM_ALLOCATION */


/*******************************************************************************
```

No consoles to display at this time.

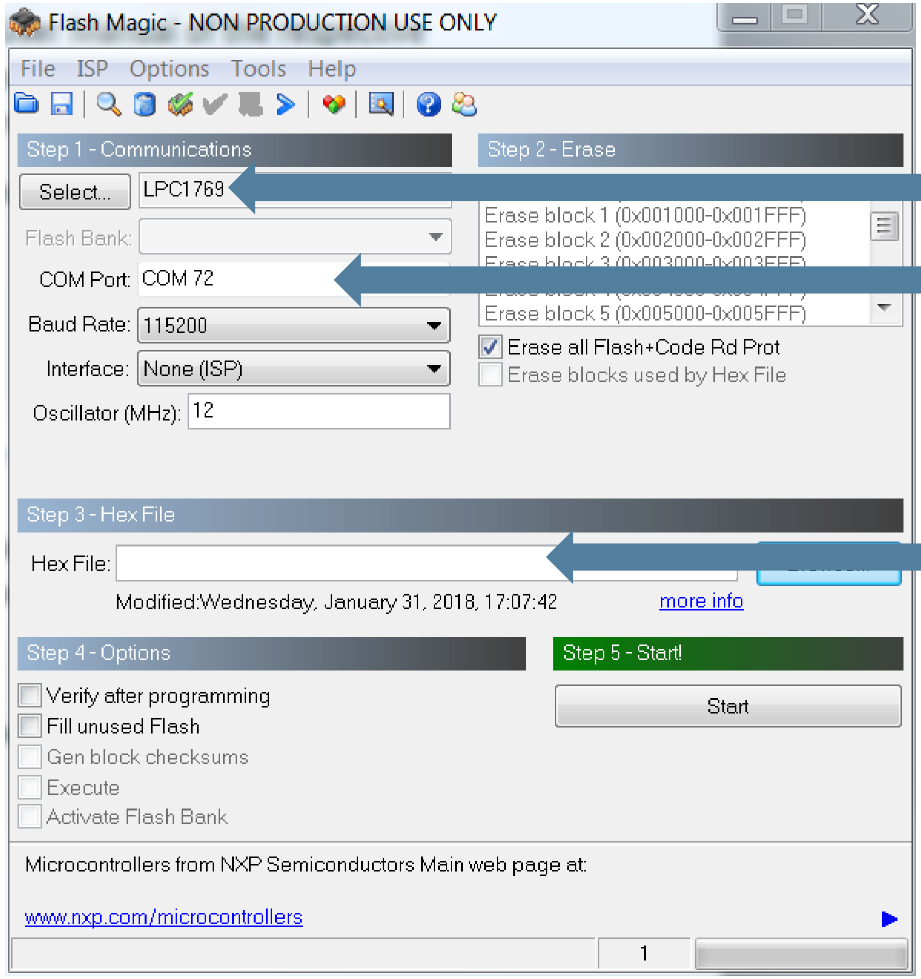
4. Build project in MCUXpresso: Create hex file



```
/*******************************************************************************
**   Main Function
*******************************************************************************/

int main (void)
{
    do
    {
        phStatus_t              status = PH_ERR_INTERNAL_ERROR;
        phNfcLib_Status_t       dwStatus;

#ifdef PH_PLATFORM_HAS_ICFRONTEND
        phNfcLib_AppContext_t AppContext = {0};
#endif /* PH_PLATFORM_HAS_ICFRONTEND */

#ifndef PH_OSAL_NULLOS
        phOsal_ThreadObj_t BasicDisc;
#endif /* PH_OSAL_NULLOS */

        /* Perform Controller specific initialization. */
        phApp_CPU_Init();

        /* Perform OSAL Initialization. */
        _Init();
```

5. Flash the MCU image

Flash Magic



Flash Magic - NON PRODUCTION USE ONLY

Step 1 - Communications

Select... LPC1769

Flash Bank:

COM Port: COM 72

Baud Rate: 115200

Interface: None (ISP)

Oscillator (MHz): 12

Step 2 - Erase

Erase block 1 (0x001000-0x001FFF)
Erase block 2 (0x002000-0x002FFF)
Erase block 3 (0x003000-0x003FFF)
Erase block 5 (0x005000-0x005FFF)

☑ Erase all Flash+Code Rd Prot
□ Erase blocks used by Hex File

Step 3 - Hex File

Hex File:

Modified:Wednesday, January 31, 2018, 17:07:42 more info

Step 4 - Options

□ Verify after programming
□ Fill unused Flash
□ Gen block checksums
□ Execute
□ Activate Flash Bank

Step 5 - Start!

Microcontrollers from NXP Semiconductors Main web page at:

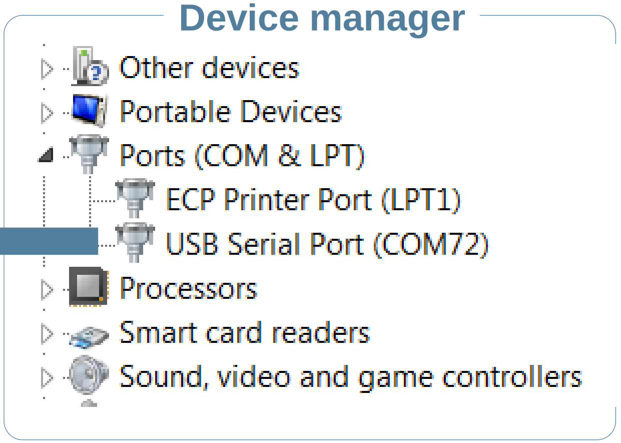
www.nxp.com/microcontrollers

1

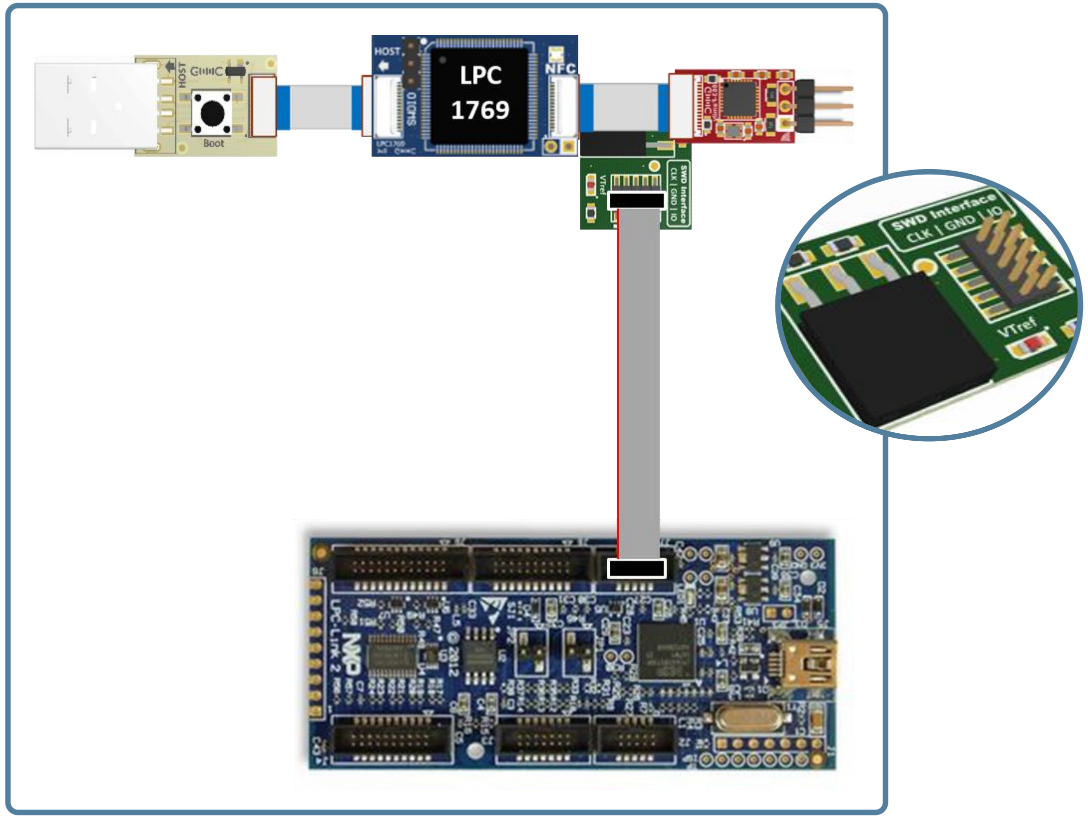
MCU

Hex image

Device manager

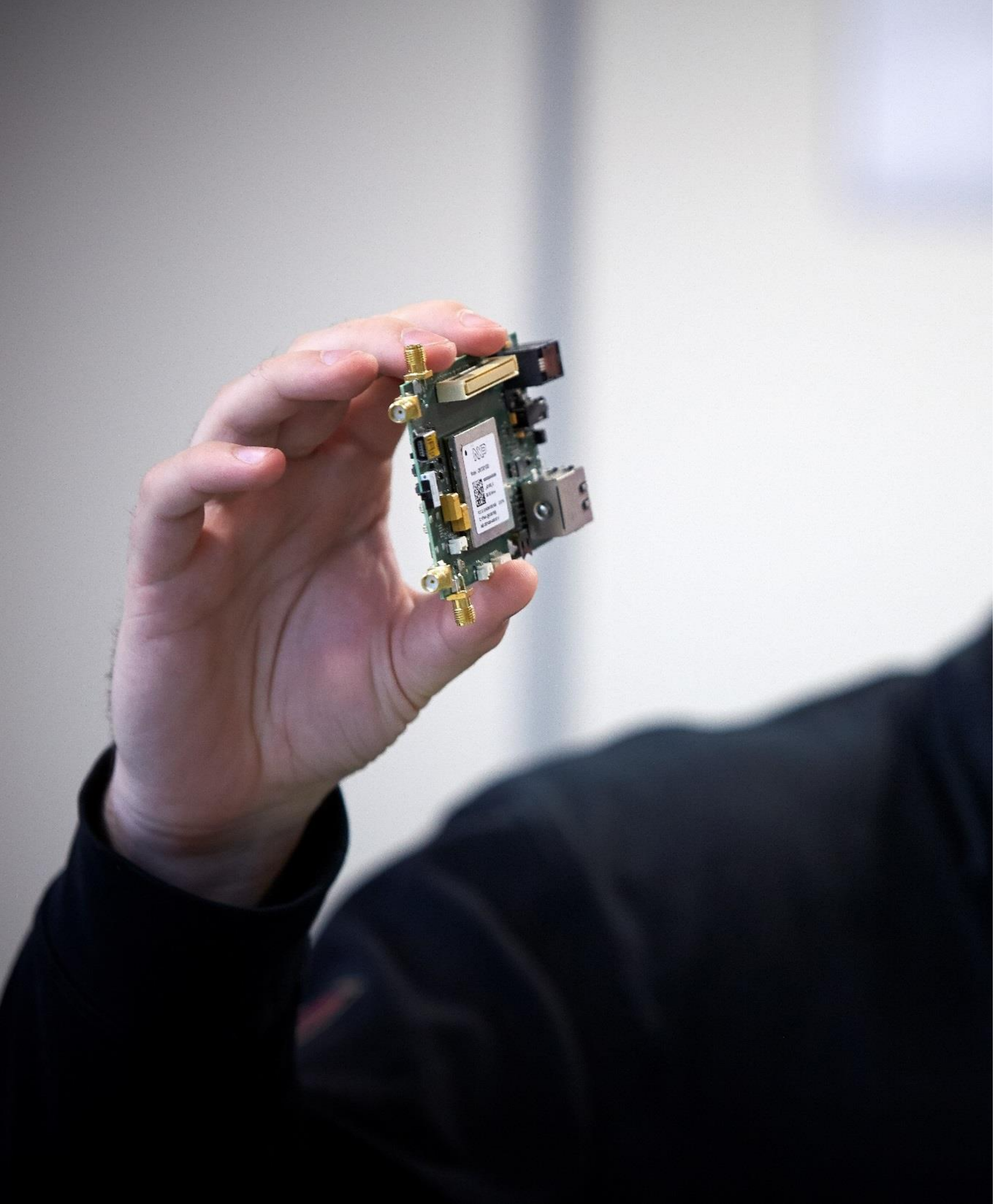


Additional tool: Debugging module



- Compatible with LPC1769 and LPC11u68, both included in the Nutshell Kit
- SWD-LPC LINK2 adapter, standard protocol for debugging
- Requires pre-programmed USB Bootloader and specific user application

EVERYTHING YOU NEED TO BUILD YOUR **NFC CONSUMABLE AUTHENTICATION** SOLUTION IS HERE!



Summary of available resources

Tags:

- NTAG 413 DNA
- NTAG 210µ
- NTAG 213 TT
- ICODE DNA

Readers:

- MFRC630 plus
- SLRC610 plus

AN11350 NTAG Originality Signature Validation
(Requires registration)

GMMC

NFC for consumables and accessories

Jordi Jofre (Speaker)
Angela Gemio (Host)

Thank you for your kind attention!

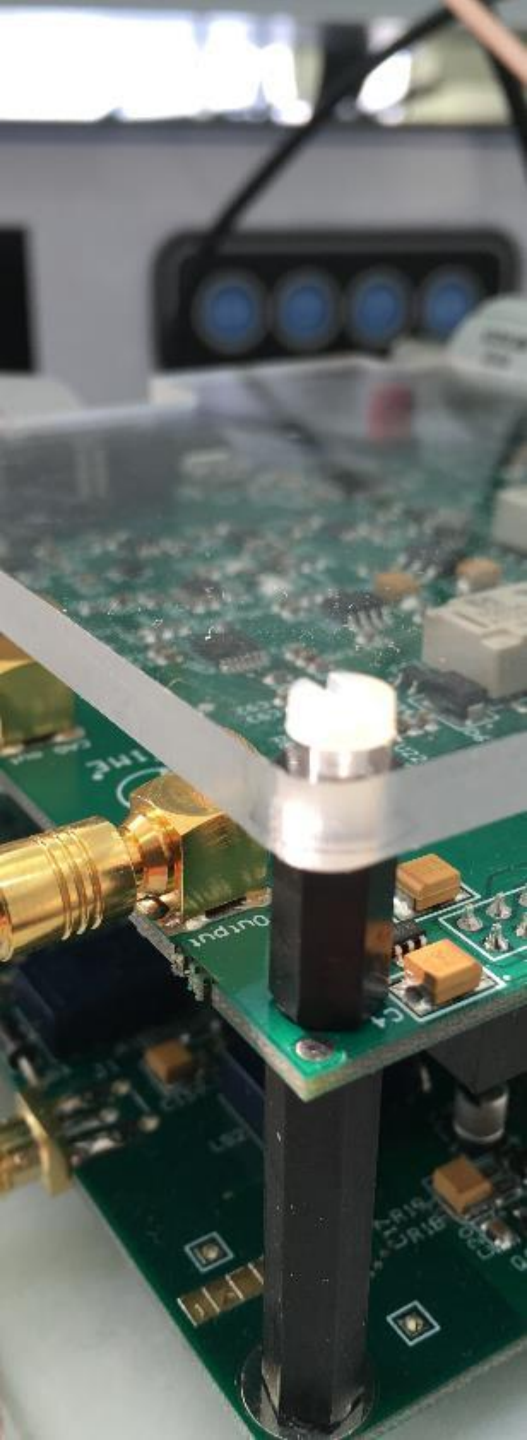
Please remember to fill out our **evaluation survey** (pop-up)

Check your email for **material download** and on-demand **video** addresses

Please check NXP and MobileKnowledge websites for **upcoming webinars** and **training sessions**

http://www.nxp.com/support/classroom-training-events:CLASSROOM-TRAINING-EVENTS

www.themobileknowledge.com/content/knowledge-catalog-0



Mobile Knowledge

Secure hardware design

Embedded software development

NFC antenna design and **evaluation**

EMV L1 pre-certification support

Mobile and **cloud application development**

Secure e2e system design

Advanced **technical training**

www.themobileknowledge.com

mk@themobileknowledge.com

MIFARE Accredited Training Partner

Approved Engineering Consultant NXP

MOBILE WORLD CAPITAL™ BARCELONA

We help companies leverage the **secure IoT revolution**