AN13288

NXP-NCI2.0 MCUXpresso examples guide

Rev. 1.4 — 2 May 2023

Application note

Document Information

Information	Content
Keywords	PN7160, NCI 2.0, NullOS, FreeRTOS, NFC, MCUXpresso, LPC, i.MX RT
Abstract	This document intents to provide a description of the PN7160 MCUXpresso examples. This project demonstrates simple integration of PN7160 NFC controller without any OS resources dependencies.

Revision history

Revision history

Rev	Date	Description
1.4.	20230502	Updated Section 7, URL added in Section 4.4
1.3	20230427	Updated Section 7, added Section 4.4 and Section 3.1
1.2	20210913	Security status changed into "Public", no content change
1.1	20210825	OM13071 Virtual COM port baud rate aligned with others HW setup in Section 3
1.0	20210706	Initial version

1 Introduction

The PN7160 NXP-NCI2.0 MCUXpresso examples shows how to easily interact with NCI-based NXP's PN7160 NFC controller in order to provide NFC capability to an embedded system with no OS resources required.

The code example is delivered in the form of MCUXpresso projects running on NXP's LPC82x, LPC55S6x microcontrollers from the LPC family and i.MX RT1170 from the Crossover Processors family.

The present example demonstrates NFC functionalities:

- R/W mode:
  - extract NDEF content from a remote NFC Forum tag (Tag Types 1 to 5) and from MIFARE Classic card
  - write NDEF content to NFC Forum Type 2, Type 4 and Type 5 tag and to MIFARE Classic card
  - authenticate/read/write with MIFARE Classic card
  - raw card access (ISO14443-3A, ISO14443-4 and ISO15693 cards)
  - multiple tags support (up to 2 of the same technology or multiprotocol card)
- P2P mode: exchange (in both way) NDEF content with remote P2P device
- Card emulation mode:
  - expose NDEF content and allow update to/from remote NFC reader (Type 4 tag emulation)
  - raw card emulation (ISO14443-4 emulation)

In this document, the terms „MIFARE Classic card“, "MIFARE DESFire card", "MIFARE card" refer to a MIFARE Classic or MIFARE DESFire IC-based contactless card.

Example of scenario with MIFARE DESFire card is available on-demand under NDA from your NXP representative.

2 HW setup

2.1 LPC82x

To set up the project, OM13071 LPCXpresso824-MAX board (http://www.nxp.com/demoboard/OM13071) is used.

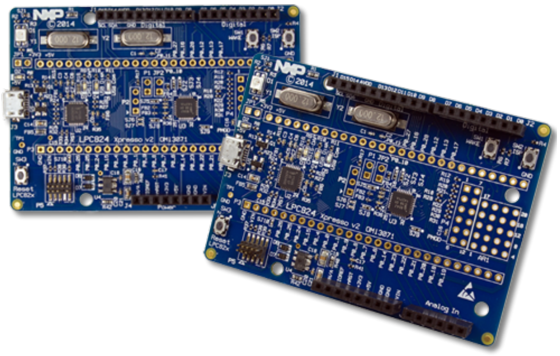


Figure 1. OM13071 LPCXpresso824-MAX board

The board must be connected to NFC controller board using the following instructions:

Table 1. OM13071 HW setup instructions

OM13071 board pin		NFC controller board signal
V_{OUT} 3.3 V	<->	VDD(PAD)
+5 V USB out	<->	VBAT and VDD(UP)
PIO0.13	<->	IRQ
PIO0.16	<->	DWL_REQ
PIO0.17	<->	VEN
GND	<->	GND
PIO0.10 / I2C0_SCL	<- for I^2C PN7160 variant->	I2C_SCL
PIO0.11/ I2C0_SDA	<- for I^2C PN7160 variant->	I2C_SDA
PIO0.15 / SPI0_SSEL0	<- for SPI PN7160 variant->	SPI_NSS
PIO0.24 / SPI0_SCK	<- for SPI PN7160 variant->	SPI_SCK
PIO0.25 / SPI0_MISO	<- for SPI PN7160 variant->	SPI_MISO
PIO0.26 / SPI0_MOSI	<- for SPI PN7160 variant->	SPI_MOSI

This matches the Arduino version of OM27160A1EVK (I^2C variant) and OM27160B1EVK (SPI variant). Those kits can then be plugged on OM13071 board to run the example.

2.2 LPC55S6x

To set up the project, LPCXpresso55S69 development board (http://www.nxp.com/demoboard/LPC55S69-EVK) is used.

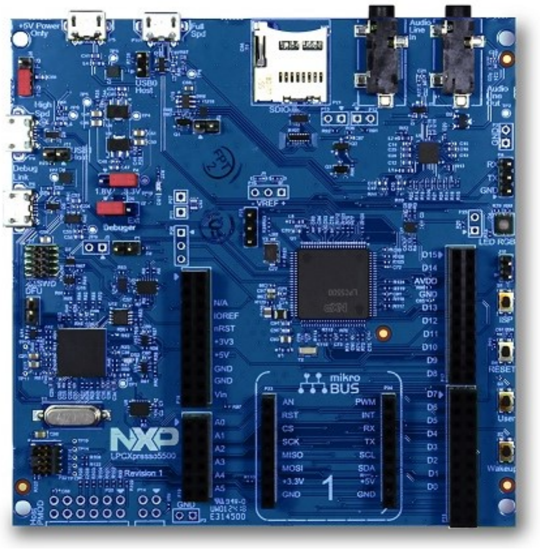


Figure 2. LPC55S69-EVK: LPCXpresso55S69 development board

The board must be connected to NFC controller board using the following instructions:

Table 2. LPC55S69-EVK HW setup instructions

LPC55S69-EVK board pin		NFC controller board signal
VDD_TARGET	<->	VDD(PAD)
+5 V	<->	VBAT and VDD(UP)
PIO1_8 / D8	<->	IRQ
PIO1_9 / D7	<->	VEN
PIO1_10 / D6	<->	DWL_REQ
GND	<->	GND
FC4_TXD_SCL_MISO_WS / D15	<- for I^{2}C PN7160 variant->	I2C_SCL
FC4_RXD_SDA_MOSI_DATA / D14	<- for I^{2}C PN7160 variant->	I2C_SDA
HS_SPI_SCK / D13	<- for SPI PN7160 variant->	SPI_SCK
HS_SPI_MISO / D12	<- for SPI PN7160 variant->	SPI_MISO
HS_SPI_MOSI / D11	<- for SPI PN7160 variant->	SPI_MOSI
HS_SPI_SSEL1 / D10	<- for SPI PN7160 variant->	SPI_NSS

This matches the Arduino version of OM27160A1EVK (I^{2}C variant) and OM27160B1EVK (SPI variant). Those kits can then be plugged on OM13071 board to run the example.

2.3 i.MX RT1170

To set up the project, i.MX RT1170 evaluation kit (http://www.nxp.com/demoboard/MIMXRT1170-EVK) is used.

Figure 3. MIMXRT1170-EVK: i.MX RT1170 evaluation kit

The board must be connected to NFC controller board using the following instructions:

Table 3. iMXRT1170 EVK HW setup instructions

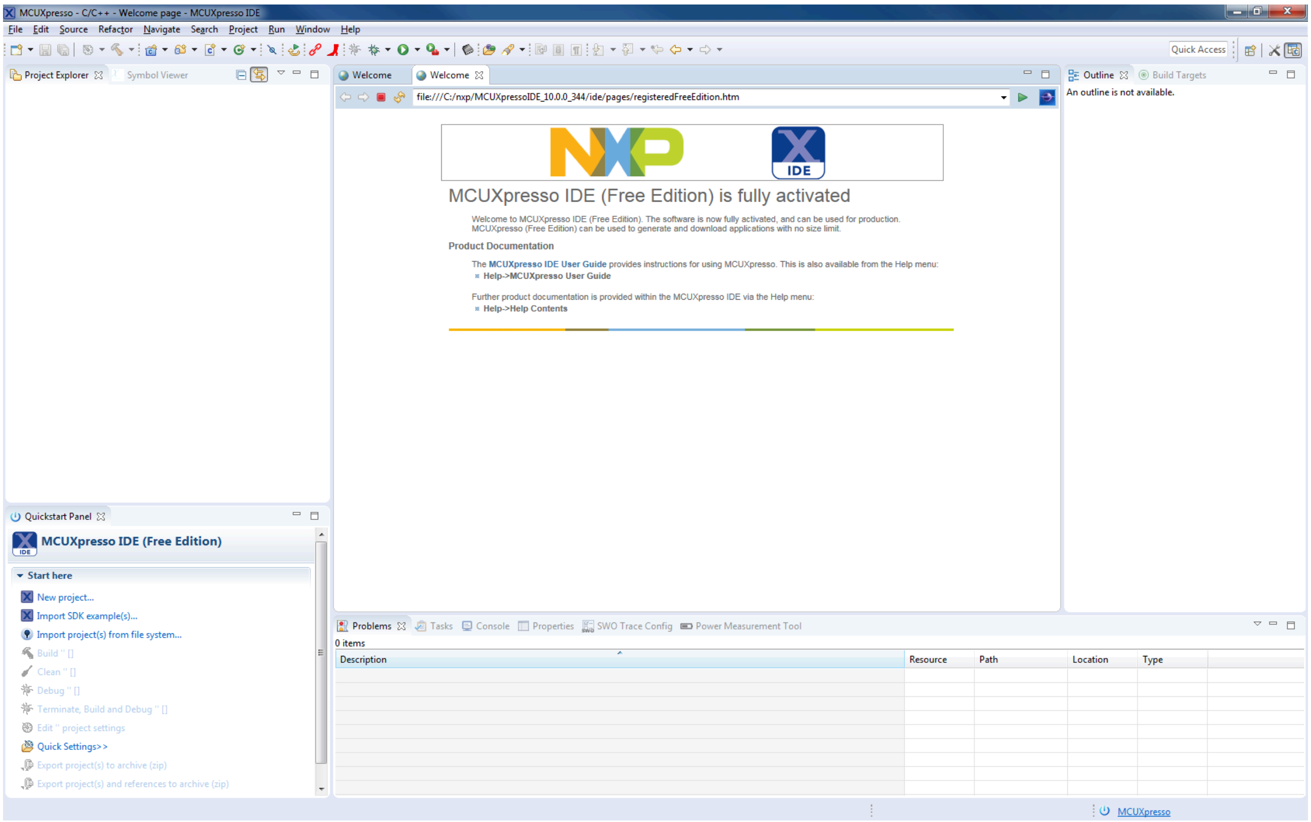
iMXRT1170 EVK board pin		NFC controller board signal
3V3	<->	VDD(PAD)
5 V	<->	VBAT and VDD(UP)
GPIO_AD_07 / D8	<->	IRQ
GPIO_AD_14 / D7	<->	VEN
GPIO_AD_00 / D6	<->	DWL_REQ
GND	<->	GND
GPIO_LPSR_05 / D15	<- for I^2C PN7160 variant->	I2C_SCL
GPIO_LPSR_04 / D14	<- for I^2C PN7160 variant->	I2C_SDA
GPIO_AD_28 / D13	<- for SPI PN7160 variant->	SPI_SCK
GPIO_AD_31 / D12	<- for SPI PN7160 variant->	SPI_MISO
GPIO_AD_30 / D11	<- for SPI PN7160 variant->	SPI_MOSI
GPIO_AD_29 / D10	<- for SPI PN7160 variant->	SPI_NSS

This matches the Arduino version of OM27160A1EVK (I^2C variant) and OM27160B1EVK (SPI variant). Those kits can then be plugged on MIMXRT1170-EVK board to run the example.

3 SW setup

MCUXpresso IDE can be downloaded from https://mcuxpresso.nxp.com/.

- Create an empty workplace in MCUXpresso IDE:



MCUXpresso - C/C++ - Welcome page - MCUXpresso IDE

file:///C:/nxp/MCUXpressoIDE_10.0.0_344/ide/pages/registeredFreeEdition.htm

MCUXpresso IDE (Free Edition) is fully activated

Welcome to MCUXpresso IDE (Free Edition). The software is now fully activated, and can be used for production. MCUXpresso (Free Edition) can be used to generate and download applications with no size limit.

Product Documentation

The **MCUXpresso IDE User Guide** provides instructions for using MCUXpresso. This is also available from the Help menu:

- **Help->MCUXpresso User Guide**

Further product documentation is provided within the MCUXpresso IDE via the Help menu:

- **Help->Help Contents**

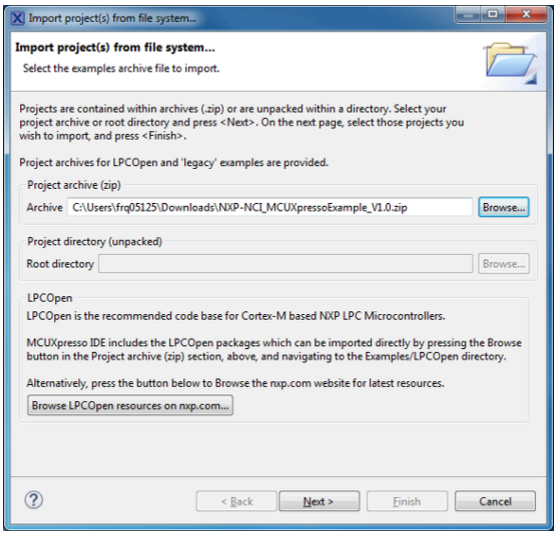
An outline is not available.

0 items

Description	Resource	Path	Location	Type

Figure 4. MCUXpresso IDE workplace

- Import the targeted project from the NXP-NCI2.0 MCUXpresso examples zip file (SW6705.zip file can be downloaded from https://www.nxp.com/doc/SW6705):



Import project(s) from file system...

Select the examples archive file to import.

Projects are contained within archives (.zip) or are unpacked within a directory. Select your project archive or root directory and press <Next>. On the next page, select those projects you wish to import, and press <Finish>.

Project archives for LPCOpen and 'legacy' examples are provided.

Project archive (zip)

Archive: C:\Users\frq05125\Downloads\NXP-NCI_MCUXpressoExample_V1.0.zip

Project directory (unpacked)

Root directory:

LPCOpen

LPCOpen is the recommended code base for Cortex-M based NXP LPC Microcontrollers.

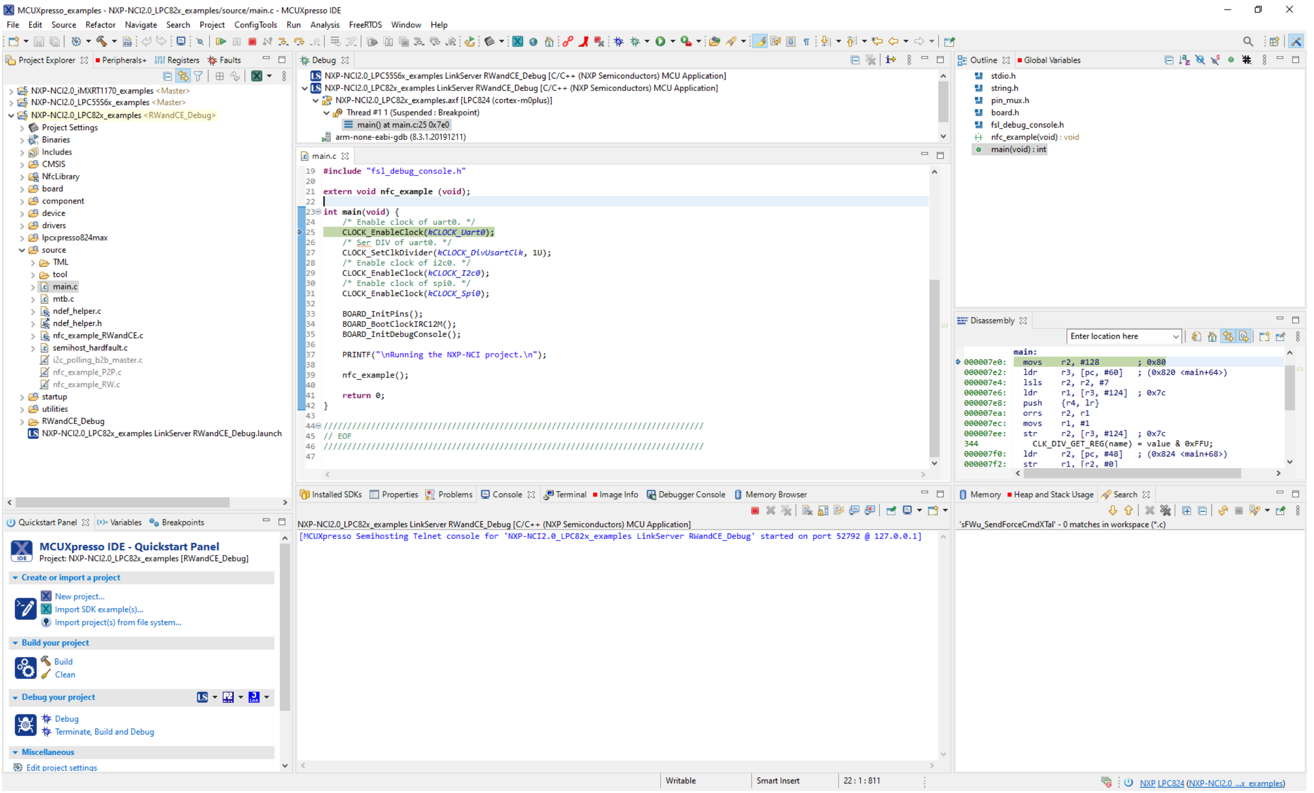
MCUXpresso IDE includes the LPCOpen packages which can be imported directly by pressing the Browse button in the Project archive (zip) section, above, and navigating to the Examples/LPCOpen directory.

Alternatively, press the button below to Browse the nxp.com website for latest resources.

Import project(s) from file system...
Browse...
Browse...
Browse LPCOpen resources on nxp.com...
< Back
Next >
Finish
Cancel

Figure 5. Importing project in MCUXpresso IDE

- Click on the “dark blue bug” icon to build the project, flash the binary into the MCU memory and start debugging:



Debug:

```
NXP-NCI2.0_LPC55S6x_examples LinkServer RWandCE_Debug [C/C++ (NXP Semiconductors) MCU Application]
NXP-NCI2.0_LPC82x_examples LinkServer RWandCE_Debug [C/C++ (NXP Semiconductors) MCU Application]
  NXP-NCI2.0_LPC82x_examples.axf [LPC824 (cortex-m0plus)]
    Thread #1 1 (Suspended : Breakpoint)
      main() at main.c:25 0x7e0
  arm-none-eabi-gdb (8.3.1.20191211)
```

main.c:

```
#include "fsl_debug_console.h"

extern void nfc_example (void);

int main(void) {
    /* Enable clock of uart0. */
    CLOCK_EnableClock(kCLOCK_Uart0);
    /* Ser DIV of uart0. */
    CLOCK_SetClkDivider(kCLOCK_DivUsartClk, 1U);
    /* Enable clock of i2c0. */
    CLOCK_EnableClock(kCLOCK_I2c0);
    /* Enable clock of spi0. */
    CLOCK_EnableClock(kCLOCK_Spi0);

    BOARD_InitPins();
    BOARD_BootClockIRC12M();
    BOARD_InitDebugConsole();

    PRINTF("\nRunning the NXP-NCI project.\n");

    nfc_example();

    return 0;
}

////////////////////////////////////////////////////////////////////////////////
// EOF
////////////////////////////////////////////////////////////////////////////////
```

Disassembly:

```
          main:
000007e0:   movs    r2, #128        ; 0x80
000007e2:   ldr     r3, [pc, #60]   ; (0x820 <main+64>)
000007e4:   lsls    r2, r2, #7
000007e6:   ldr     r1, [r3, #124]  ; 0x7c
000007e8:   push    {r4, lr}
000007ea:   orrs    r2, r1
000007ec:   movs    r1, #1
000007ee:   str     r2, [r3, #124]  ; 0x7c
344           CLK_DIV_GET_REG(name) = value & 0xFFU;
000007f0:   ldr     r2, [pc, #48]   ; (0x824 <main+68>)
000007f2:   str     r1, [r2, #0]
```

Console:

```
NXP-NCI2.0_LPC82x_examples LinkServer RWandCE_Debug [C/C++ (NXP Semiconductors) MCU Application]
[MCUXpresso Semihosting Telnet console for 'NXP-NCI2.0_LPC82x_examples LinkServer RWandCE_Debug' started on port 52792 @ 127.0.0.1]
```

Search: 'sFWu_SendForceCmdXTal' - 0 matches in workspace (*.c)

Figure 6. Debugging project in MCUXpresso IDE

- Open a terminal (i.e. TeraTerm, HyperTerminal, Putty …) to the virtual COM port with the following configuration: baud rate=115200, 8 data bits, no parity, 1 stop bit, no flow control
  Related port number can be retrieved from the “Ports (COM & LPT)” list inside computer “Device Manager”:

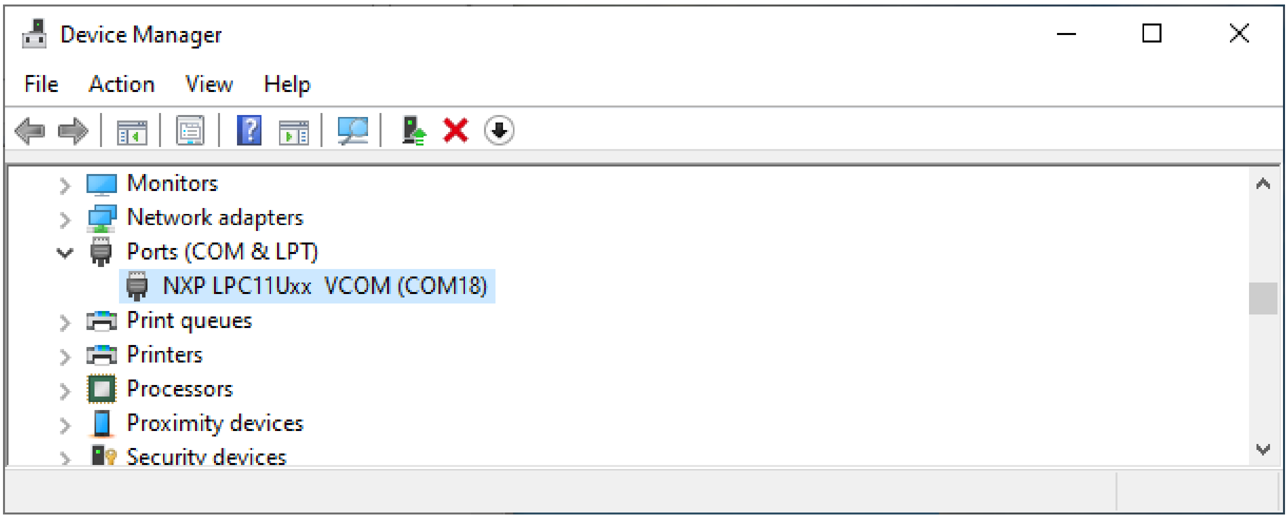
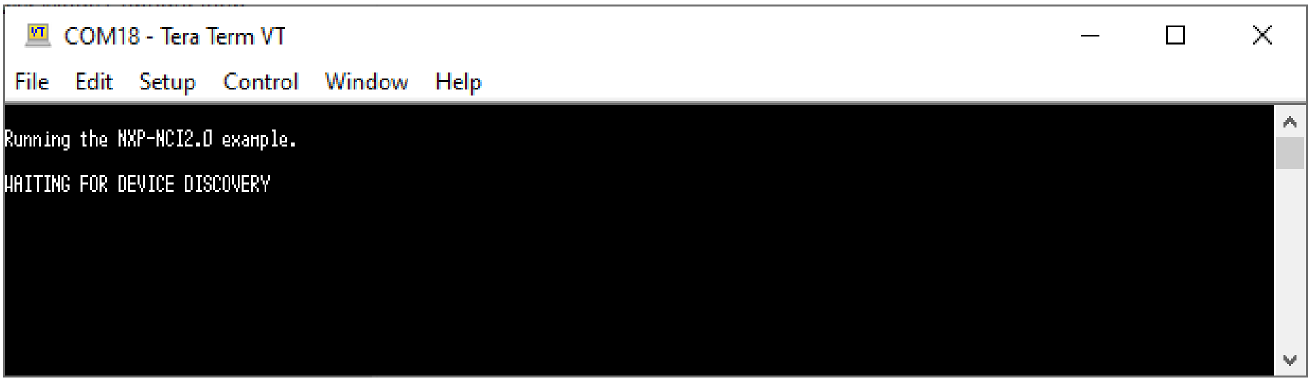


Figure 7. Retrieving COM port number from Device Manager

- Start the execution (clicking on « Resume » button or pressing ‘f8’). This launches the default demo application and following message is displayed in the terminal window:



```
Running the NXP-NCI2.0 example.

WAITING FOR DEVICE DISCOVERY
```

Figure 8. External Terminal output

3.1 Switching between examples

To switch between examples, click to "hammer" button (check Figure 9) and select example.

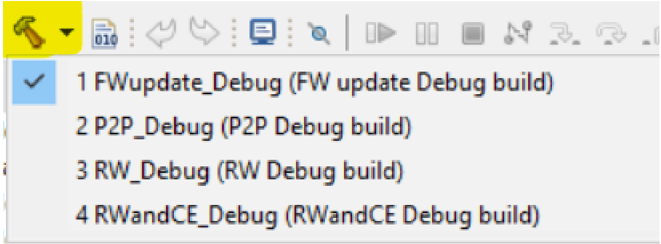

1 FWupdate_Debug (FW update Debug build)
2 P2P_Debug (P2P Debug build)
3 RW_Debug (RW Debug build)
4 RWandCE_Debug (RWandCE Debug build)

Figure 9. Switching between examples

4 Demonstration

NXP-NCI2.0 MCUXpresso examples projects include 3 possible configurations:

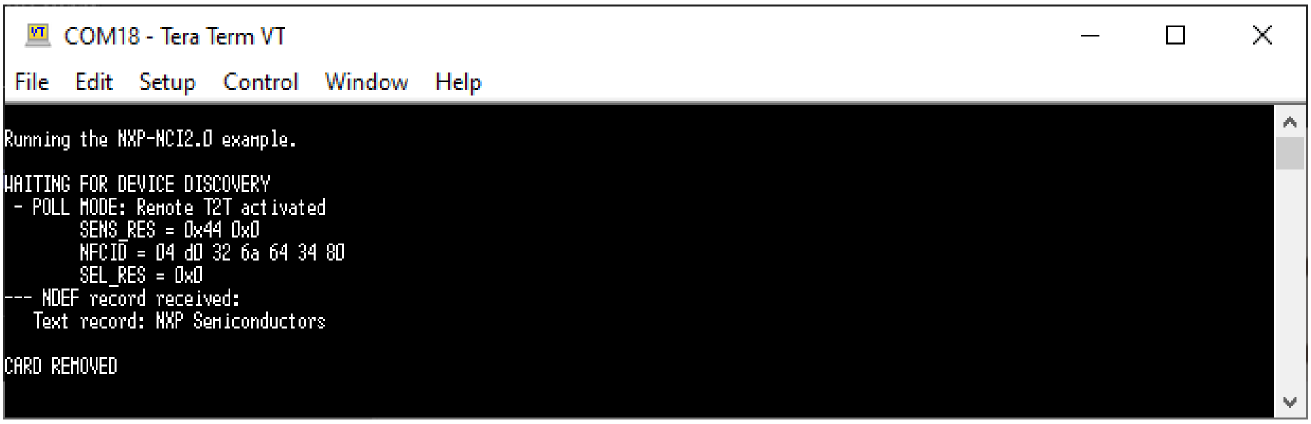
- *RWandCE*: demonstrates reader mode (extraction of NDEF content from remote NFC cards) and card emulation (exposing NDEF content to remote NFC reader)
- *RW*: demonstrates raw communication with ISO14443-3A, ISO14443-4, ISO15693 and MIFARE cards
- *P2P*: demonstrates P2P communication (sending NDEF content and receiving NDEF Content) with remote NFC P2P device

The configuration by default is *RWandCE*, it can be selected when building or debugging.

A 4th example configuration is available within LPC55S6x MCUXpresso project. Purpose is to demonstrate PN7160 FW update driven from an MCU. Detailed information is further provided in Section 7.

4.1 RWandCE

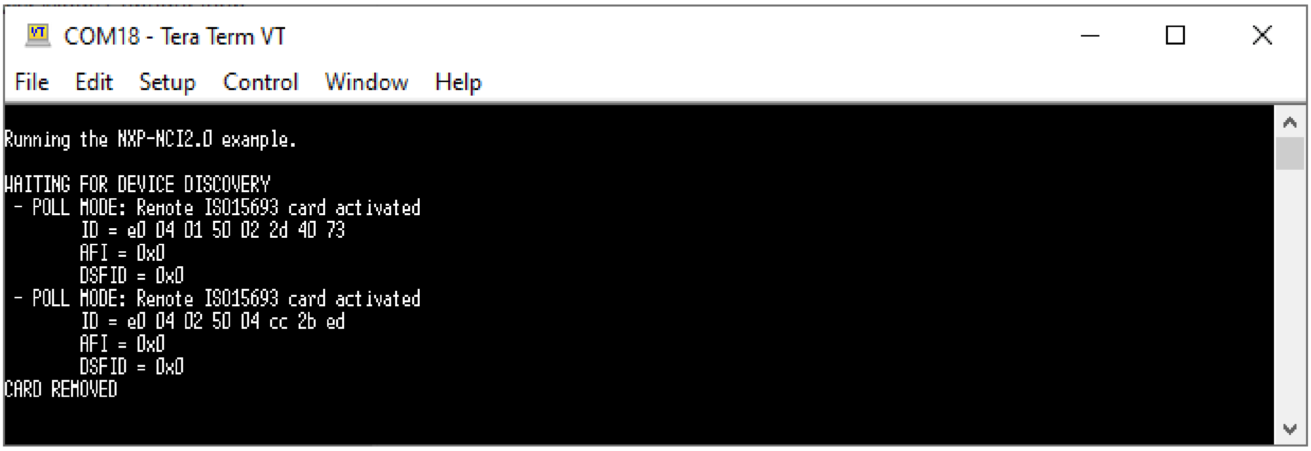
Bringing an NFC Forum Tag containing NDEF content leads to a message display in the terminal window (in below example a Type 2 tag containing Text type NDEF message “NXP Semiconductors”):



```
Running the NXP-NCI2.0 example.

WAITING FOR DEVICE DISCOVERY
 - POLL MODE: Remote T2T activated
        SENS_RES = 0x44 0x0
        NFCID = 04 d0 32 6a 64 34 80
        SEL_RES = 0x0
--- NDEF record received:
   Text record: NXP Semiconductors

CARD REMOVED
```

Figure 10. Terminal output when NDEF tag is read

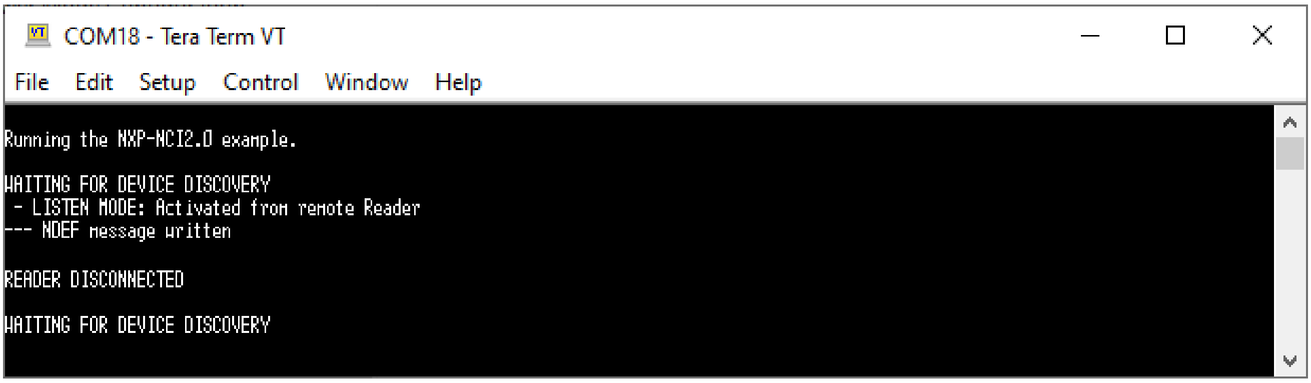
In case of several tags, the related information will be displayed one after the other in such way:



```
Running the NXP-NCI2.0 example.

WAITING FOR DEVICE DISCOVERY
 - POLL MODE: Remote ISO15693 card activated
        ID = e0 04 01 50 02 2d 40 73
        AFI = 0x0
        DSFID = 0x0
 - POLL MODE: Remote ISO15693 card activated
        ID = e0 04 02 50 04 cc 2b ed
        AFI = 0x0
        DSFID = 0x0
CARD REMOVED
```

Figure 11. Terminal output when 2 tags are detected

Approaching an NFC reader (for instance an Android NFC phone running NXP TagInfo application) will give the following output:



```
Running the NXP-NCI2.0 example.

WAITING FOR DEVICE DISCOVERY
 - LISTEN MODE: Activated from remote Reader
--- NDEF message written

READER DISCONNECTED

WAITING FOR DEVICE DISCOVERY
```

Figure 12. Terminal output when exposing NDEF content

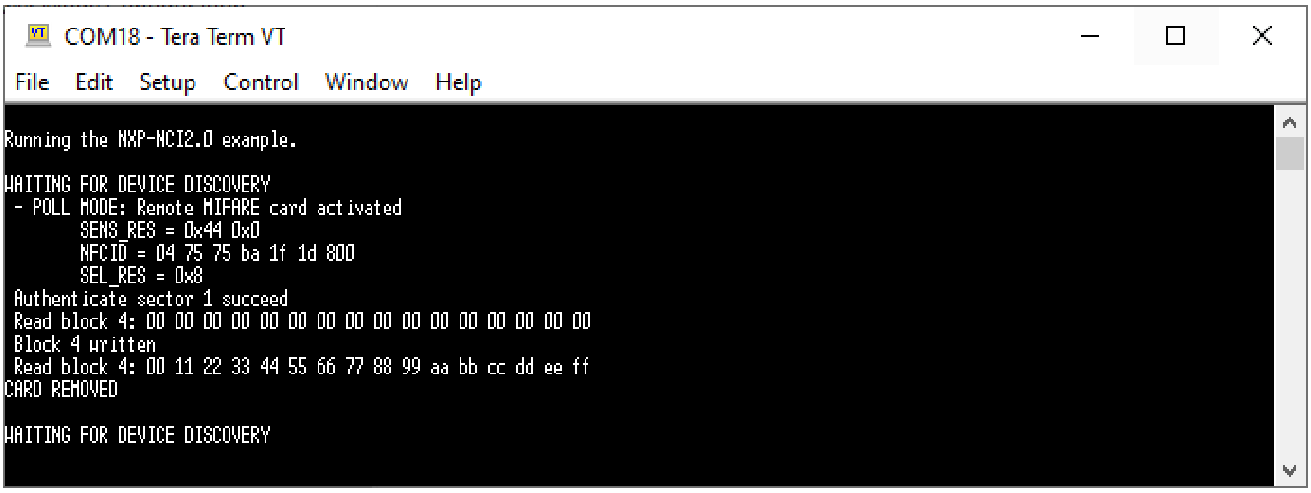
4.2 RW

When running this configuration, the example shows dedicated scenario according to the card detected:

- ISO14443-3A: read, write and read back of a memory block
- ISO14443-4: ISO7816-4 APDU exchange
- ISO15693: read, write and read back of a memory block
- MIFARE Classic: Authenticate then read, write and read back of a memory block

authenticationcard memory read, write, then read back.

For instance, bringing a MIFARE card reader gives such output:

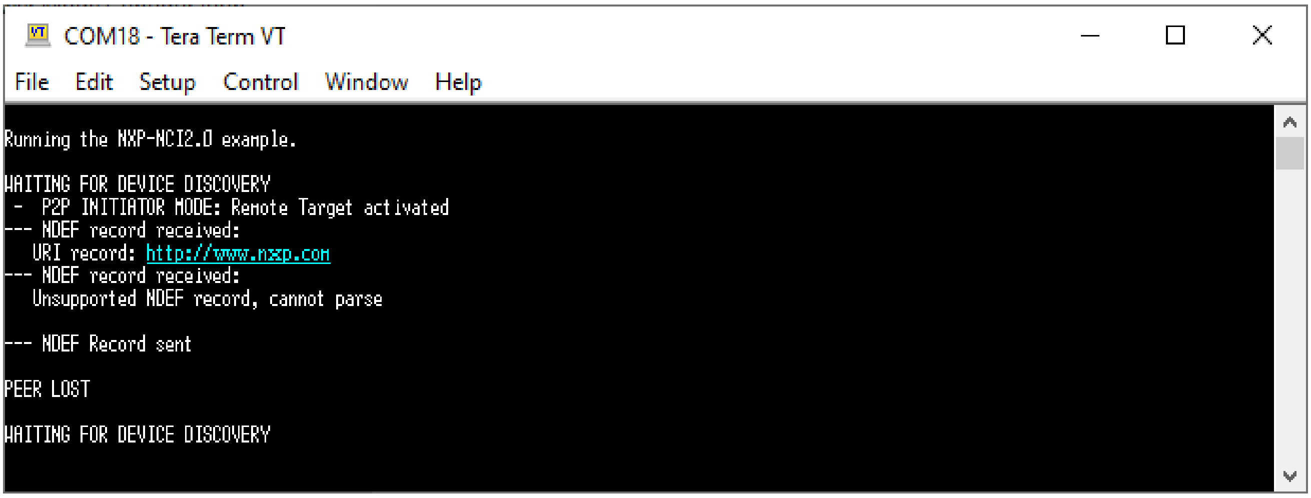


```
Running the NXP-NCI2.0 example.

WAITING FOR DEVICE DISCOVERY
 - POLL MODE: Remote MIFARE card activated
        SENS_RES = 0x44 0x0
        NFCID = 04 75 75 ba 1f 1d 80
        SEL_RES = 0x8
 Authenticate sector 1 succeed
 Read block 4: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 Block 4 written
 Read block 4: 00 11 22 33 44 55 66 77 88 99 aa bb cc dd ee ff
CARD REMOVED

WAITING FOR DEVICE DISCOVERY
```

Figure 13. Terminal output when communicating with MIFARE Card in raw mode

4.3 P2P mode

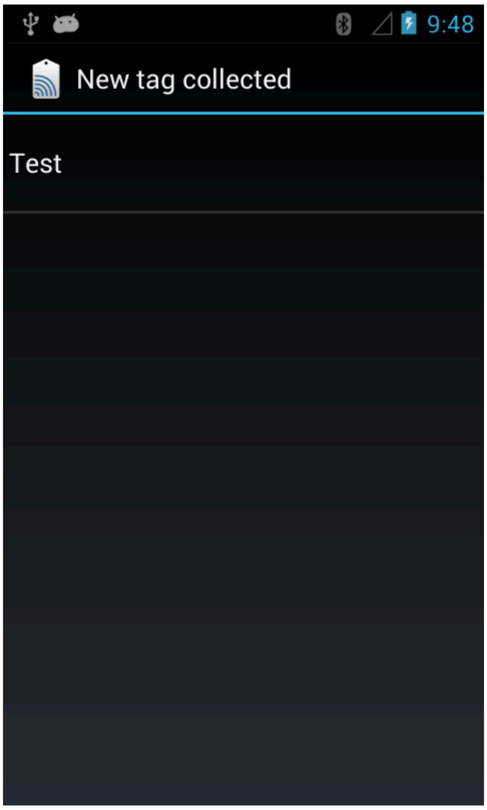
Bringing an NFC Android phone and « beaming » a URL (select the URL inside the phone web browser, tap the phone to the antenna then click of the screen when invited for it by the Android « Beam » service) gives such result:



```
Running the NXP-NCI2.0 example.

WAITING FOR DEVICE DISCOVERY
 -  P2P INITIATOR MODE: Remote Target activated
--- NDEF record received:
   URI record: http://www.nxp.com
--- NDEF record received:
   Unsupported NDEF record, cannot parse

--- NDEF Record sent

PEER LOST

WAITING FOR DEVICE DISCOVERY
```

Figure 14. Terminal output example when exchanging data with Android NFC phone

Simultaneously, the phone displays the received NDEF record from the NXP-NCI example project (NDEF Text type « Test » message):



Test

Figure 15. Android phone receiving NDEF message from NXP-NCI example project

All information provided in this document is subject to legal disclaimers.

4.4 CE Scenario 2

For more information about this scenario, check (https://www.nxp.com/doc/AN13861).

To use this example, everything what is needed is to run the example and place a phone with the TagInfo application running (or any other application for tag reading) on reader and NDEF will be read.

To store NDEF, use the TagWriter and save an NDEF to the PN7160 in CE.

5 SW description

5.1 Architecture overview

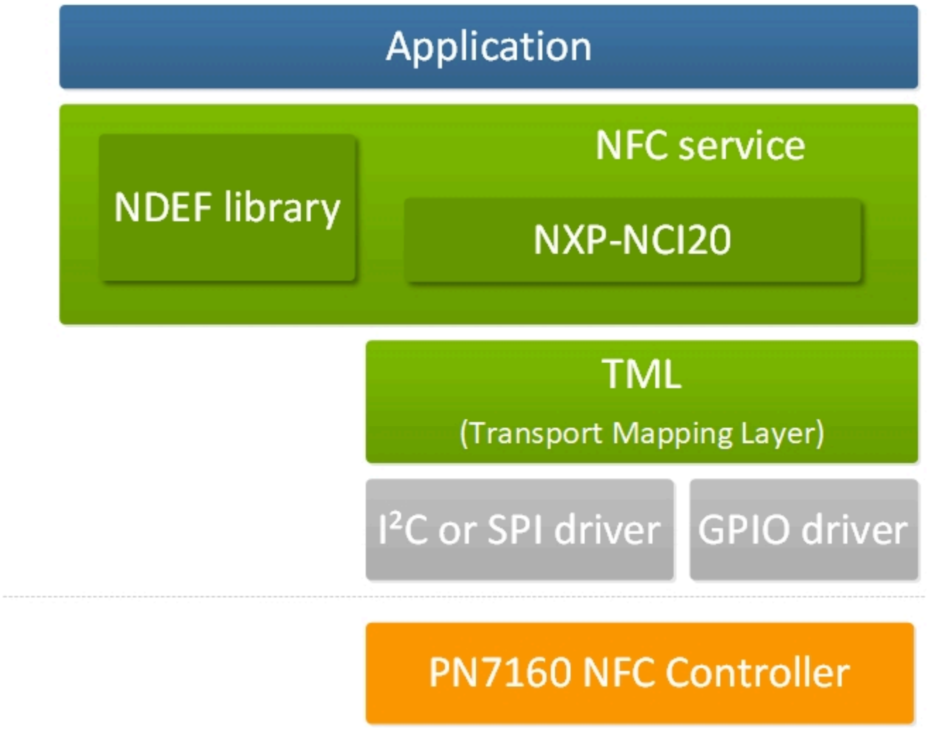


Figure 16. Example SW architecture overview

The Application consists in an NFC task using NFC library API to register for NDEF functionalities and manage NXP-NCI processing.

{NXP-NCI} module offers high-level NFC API:

- Connection and configuration of the NFC controller
- Start of the NFC discovery
- Wait for NFC discovery
- Process of the NFC discovery
- Raw access to remote tag or reader discovered

{NDEF library} module is composed of independent submodule:

- {RW_NDEF} implements NDEF extraction from NFC Forum tags (all 5 NFC Forum defined tag types + MIFARE Classic card) and NDEF write to NFC Forum Type 2, Type 4, Type 5 tags and MIFARE Classic cards
- {P2P_NDEF} implements NDEF data exchange with P2P device (over NFC Forum LLCP and SNEP protocols)
- {T4T_NDEF_emu} implements NDEF message exposure through card emulation (NFC Forum Type 4 Tag protocol)

{TML} module brings HW abstraction to NFC library (abstract how the connection to NFC controller IC is managed).

5.2 Porting recommendation for other MCUs

The present code example can be easily ported to any other target providing I^2C or SPI master and GPIO capabilities.

The only module requiring adaptation is the TML component (relates to how the target provides this support), others modules being platform agnostic.

6 Example customization

6.1 SPI selection

By default the MCUXpresso projects are configured for I^{2}C host interface, fitting for OM27160A1EVK (including PN7160A1HN I^{2}C variant).

Changing the configuration to fit with OM27160B1EVK (including PN7160B1HN SPI variant) is done within *board/board.h* file, commenting out the I^{2}C definition and uncommenting SPI one:

```
/* NXPNCI interface selection: enable only one according to NFC Controller supported interface */
#define BOARD_NXPNCI_INTERFACE_I2C
//#define BOARD_NXPNCI_INTERFACE_SPI
```

6.2 I^{2}C address/speed

NFC controller I^{2}C address is by default set to 0x28 (matching OM27160A1EVK HW configuration) and speed is set to 100 kbps. It can be changed in the file *board/board.h*:

```
#define BOARD_NXPNCI_I2C_BAUDRATE  (100000)
#define BOARD_NXPNCI_I2C_ADDR      (0x28)
```

6.3 PIOs assignment

In case a different connection is used than the one described in Section 2, definition in *board/board.h* file must reflect PIOs assignment:

```
#define BOARD_NXPNCI_GPIO_PORT       (0U)
#define BOARD_NXPNCI_IRQ_PORT        (0U)
#define BOARD_NXPNCI_IRQ_PIN        (13U)
#define BOARD_NXPNCI_VEN_PORT        (0U)
#define BOARD_NXPNCI_VEN_PIN        (17U)
#define BOARD_NXPNCI_DWL_PORT        (0U)
#define BOARD_NXPNCI_DWL_PIN        (16U)
```

6.4 Discovery technologies

The list of technologies inside the discovery loop can be configured by setting `DiscoveryTechnologies` variable defined in *source/nfc_example_***XXX***.c* file.

By default, all technologies (required for the aimed demonstration) are enabled.

Here is the list of all technologies supported by PN7160:

```
MODE_POLL | TECH_PASSIVE_NFCA
MODE_POLL | TECH_PASSIVE_NFCB
MODE_POLL | TECH_PASSIVE_NFCF
MODE_POLL | TECH_PASSIVE_15693
MODE_POLL | TECH_ACTIVE_NFC
MODE_LISTEN | TECH_PASSIVE_NFCA
MODE_LISTEN | TECH_PASSIVE_NFCB
MODE_LISTEN | TECH_PASSIVE_NFCF
MODE_LISTEN | TECH_ACTIVE_NFCA
```

6.5 Settings configuration

Dedicated settings can be applied to the NXP-NCI NFC Controller. Those are configured thanks to *NfcLibrary/inc/Nfc_settings.h* file.

- NFC settings configuration

```
/* Following definitions specifies which settings will apply
 * when NxpNci_ConfigureSettings() API is called from the application
 */
#define NXP_CORE_CONF        1
#define NXP_CORE_STANDBY     1
#define NXP_CORE_CONF_EXTN   1
#define NXP_CLK_CONF         1 // 1=Xtal, 2=PLL
#define NXP_TVDD_CONF        1 // 1=CFG1, 2=CFG2
#define NXP_RF_CONF          1
```

- NXP_CORE_CONF setting definition

```
/* NCI standard dedicated settings
 * Refer to NFC Forum NCI standard for more details
 */
uint8_t NxpNci_CORE_CONF[]={0x20, 0x02, 0x05, 0x01,  /* CORE_SET_CONFIG_CMD */
    0x00, 0x02, 0xFE, 0x01                           /* TOTAL_DURATION */
};
```

- NXP_CORE_CONF_EXTN setting definition

```
/* NXP-NCI extension dedicated setting
 * Refer to NFC controller User Manual for more details
 */
uint8_t NxpNci_CORE_CONF_EXTN[]={0x20, 0x02, 0x05, 0x01,    /* CORE_SET_CONFIG_CMD */
    0xA0, 0x40, 0x01, 0x00,                                 /* TAG_DETECTOR_CFG */
};
```

- NXP_CORE_STANDBY setting definition

```
/* NXP-NCI standby enable setting
 * Refer to NFC controller User Manual for more details
 * last byte indicates enable/disable
 */
uint8_t NxpNci_CORE_STANDBY[]={0x2F, 0x00, 0x01, 0x00};
```

- NXP_CLK_CONF setting definition

```
/* NXP-NCI CLOCK configuration
 * Refer to NFC controller Hardware Design Guide document for more details
 */
 #if (NXP_CLK_CONF == 1)
 /* Xtal configuration */
 uint8_t NxpNci_CLK_CONF[]={0x20, 0x02, 0x05, 0x01, /* CORE_SET_CONFIG_CMD */
   0xA0, 0x03, 0x01, 0x08                           /* CLOCK_SEL_CFG */
 };
 #else
 /* PLL configuration */
 uint8_t NxpNci_CLK_CONF[]={0x20, 0x02, 0x09, 0x02, /* CORE_SET_CONFIG_CMD */
   0xA0, 0x03, 0x01, 0x11,                          /* CLOCK_SEL_CFG */
   0xA0, 0x04, 0x01, 0x01                           /* CLOCK_TO_CFG */
 };
 #endif
```

- NXP_TVDD_CONF setting definition

```
/* NXP-NCI TVDD configuration
 * Refer to NFC controller Hardware Design Guide document for more details
 */
 #if (NXP_TVDD_CONF == 1)
 /* CFG1: VBAT is used to generate the VDD(TX) through TXLDO */
 uint8_t NxpNci_TVDD_CONF[]={0x20, 0x02, 0x0F, 0x01, 0xA0, 0x0E, 0x0B, 0x01, 0x01, 0x01, 0x00, 0x00, 0x00, 0x1E,
 0xBB, 0x00, 0x00, 0x0C};
 #else
 /* CFG2: external 5V is used to generate the VDD(TX) through TXLDO */
 uint8_t NxpNci_TVDD_CONF[]={0x20, 0x02, 0x0F, 0x01, 0xA0, 0x0E, 0x0B, 0x01, 0x01, 0x03, 0x00, 0x00, 0x00, 0x1E,
 0xBB, 0x00, 0x00, 0x0C};
 #endif
```

- NXP_RF_CONF settings definition

```
/* NXP-NCI RF configuration
 * Refer to NFC controller Antenna Design and Tuning Guidelines document for
 * more details
 */
uint8_t NxpNci_RF_CONF[]={};
```

6.6 Shared NDEF message

NDEF message shared in P2P or card emulation mode (or even in RW mode while NDEF write operation is enabled, see Section 6.7) can be changed. Simply modify value of `NDEF_MESSAGE` variable in file *source/nfc_example_RWandCE.c* or *source/nfc_example_P2P.c*, following NFC Forum NDEF specification.

```
const char NDEF_MESSAGE[14] = {
    0xC1,                  // MB/ME/CF/1/IL/TNF
    0x01,                  // TYPE LENGTH
    0x07 >> 24,            // PAYLOAD LENTGH MSB
    0x07 >> 16,            // PAYLOAD LENTGH
    0x07 >> 8,             // PAYLOAD LENTGH
    0x07 & 0xFF,           // PAYLOAD LENTGH LSB
    'T',                   // TYPE
                           // PAYLOAD
    0x02,                  // Language length
    'e', 'n',              // Language
    'T', 'e', 's', 't' };
```

6.7 NDEF write operation

Demonstration the NDEF write operation to a remote NFC card is present in the RWandCE demo application (but not enabled by default to prevent unintentional overwriting of tag content). Enabling it is done defining `RW_NDEF_WRITING` compile flag in *source/nfc_example_RWandCE.c* file (just uncomment present definition), before building the project. Then the write operation will occur just after the NDEF read operation.

The NDEF message which is written is defined in `NDEF_MESSAGE` variable (see Section 6.6).

Only Type 2, Type 4 tags and MIFARE Classic card are currently supported by the NFC library for NDEF write operation. For others tag types, write operation will simply not occur but no issue will be reported. Furthermore, the tag must be already NDEF formatted, the NFC library not implementing NDEF formatting functionality.

6.8 P2P timing optimization

The current example implementation allows sharing in both way NDEF message with a peer device (receiving and sending an NDEF message) in P2P mode over SNEP NFC Forum protocol.

The SNEP standard protocol being also implemented as native feature of Android, so-called "Beam" service, the NXP-NCI example shows NDEF message exchanges with NFC Android devices. Unfortunately, because of the "Beam" service implementation, the Android device cannot send any NDEF message after it has received one (until a new tap occurs).

To work around this limitation, the NXP-NCI example defines a way to postone sending NDEF message after the peer discovery, to give the Android device user to "Beam" the expected content. This is implemented as `NDEF_PUSH_DELAY_COUNT` variable inside *NfcLibrary/NdefLibrary/src/P2P_NDEF.c* file.

```
/* Defines the number of symmetry exchanges is expected before initiating
 * the NDEF push (to allow a remote phone to beam an NDEF message first) */
#define NDEF_PUSH_DELAY_COUNT    2
```

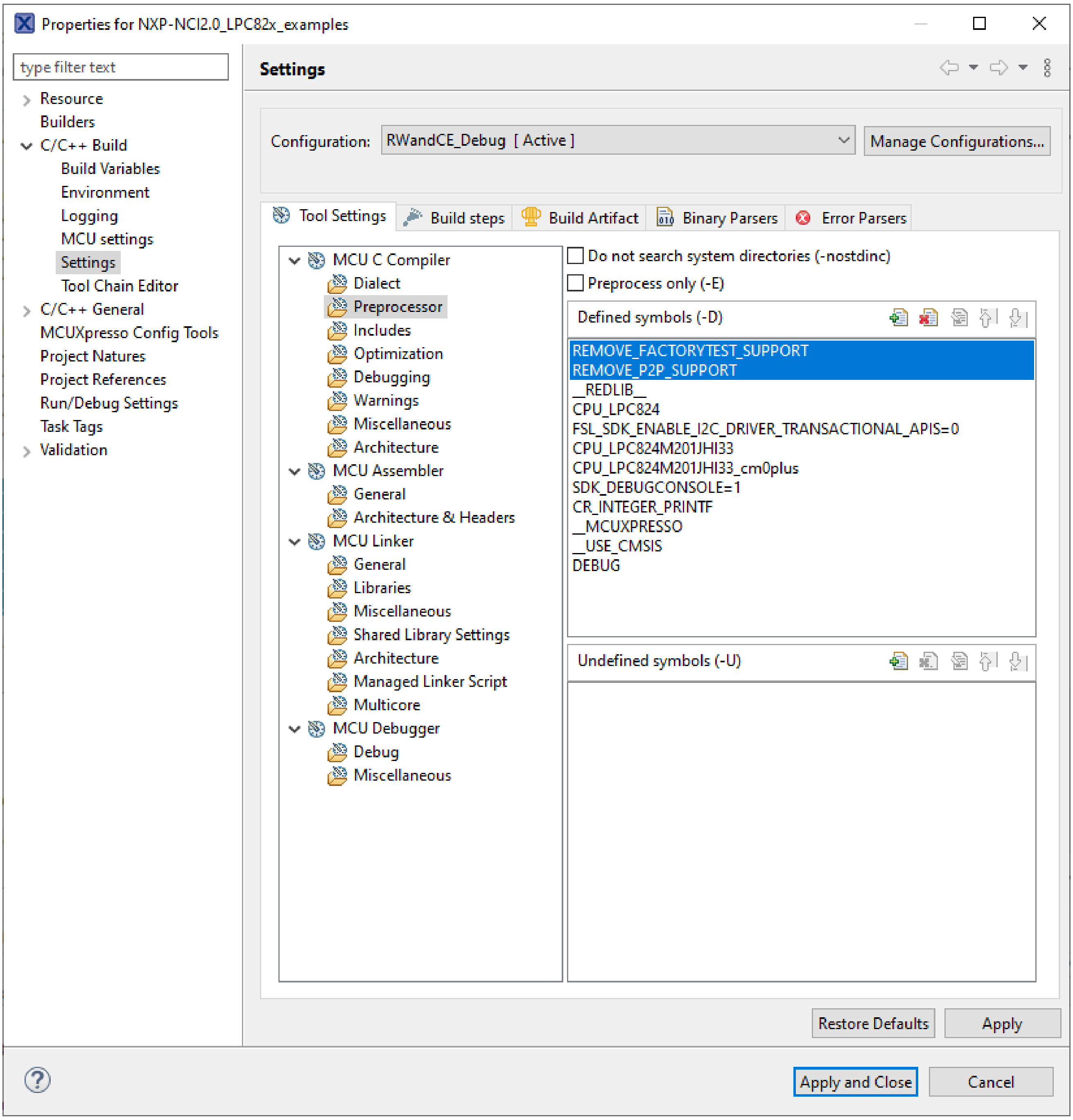
6.9 Code optimization

Compile flags exist in this SW example allowing to separately remove some features from the build, allowing to optimize the overall memory requirement whenever some features are not required by the targeted application:

- `REMOVE_FACTORYTEST_SUPPORT`: relates to dedicated factory testing features (continuous RF ON and PRBS modes)
- `REMOVE_P2P_SUPPORT`: relates to NFC P2P feature
- `REMOVE_CARDEMU_SUPPORT`: relates to the card emulation feature
- `REMOVE_RW_SUPPORT`: relates to reader/writer feature
- `REMOVE_NDEF_SUPPORT`: relates to NDEF handling for all RW, card emulation and P2P modes

List of APIs disabled by those flags can easily br retrieved by looking to *NfcLibrary/inc/Nfc.h* file.

They can be defined in the project properties to avoid embedding useless modules:



Properties for NXP-NCI2.0_LPC82x_examples

Settings

Configuration: RWandCE_Debug [Active]

C/C++ Build > Settings > Tool Settings > MCU C Compiler > Preprocessor

☐ Do not search system directories (-nostdinc)

☐ Preprocess only (-E)

Defined symbols (-D)

```
REMOVE_FACTORYTEST_SUPPORT
REMOVE_P2P_SUPPORT
__REDLIB__
CPU_LPC824
FSL_SDK_ENABLE_I2C_DRIVER_TRANSACTIONAL_APIS=0
CPU_LPC824M201JHI33
CPU_LPC824M201JHI33_cm0plus
SDK_DEBUGCONSOLE=1
CR_INTEGER_PRINTF
__MCUXPRESSO
__USE_CMSIS
DEBUG
```

Undefined symbols (-U)

Figure 17. MCUXpresso project Preprocessor definition

6.10 NCI communication debugging

Enabling NCI communication traces can be done defining `NCI_DEBUG` compile flag inside the project properties (see Figure 17), or directly in *NfcLibrary/NxpNci20/inc/NxpNci.h* file, before building the project.

Pay attention that this significantly increases overall memory requirement.

7 FW update example

This example is available as "FWupdate" configuration of LPC55S6x MCUXpresso project.

It shows how to update the PN7160 internal firmware.

The firmware to be flashed is included in *sFWupdate/phDnldNfc_UpdateSeq.c*.

```
uint8_t gphDnldNfc_DlSequence[] = {
0x00, 0xE4, 0xC0, 0x00, 0x05, 0x50, 0x1C, 0xCB, 0x17, 0xCD,
0xDD, 0x2A, 0xDB, 0xB9, 0xC0, 0xBF, 0xE1, 0x5D, 0x7B, 0xFA,
0xD6, 0x4A, 0x53, 0xDA, 0x23, 0x45, 0x3E, 0x18, 0x9C, 0xA9,
0xA7, 0xDB, 0x2E, 0x64, 0x0C, 0xE0, 0x9B, 0x9E, 0xD1, 0xE4,
0xE1, 0x53, 0x26, 0xFA, 0x71, 0x22, 0xA8, 0xE3, 0xFA, 0xC8,
0x2F, 0x94, 0x9F, 0xCD, 0xA5, 0x6D, 0x52, 0x2F, 0x53, 0x06,
0xD2, 0x86, 0x47, 0xD3, 0x08, 0xFE, 0x7F, 0x97, 0xA1, 0x78,
0x88, 0x14, 0xA2, 0x02, 0xD2, 0xE4, 0xE0, 0x15, 0x1B, 0x9F,
0xC0, 0x12, 0x6A, 0x8B, 0x90, 0x03, 0x50, 0x12, 0xCB, 0x2D,
0x3B, 0xF0, 0x98, 0xE1, 0xD8, 0x12, 0x64, 0xF1, 0x9B, 0x96,
0x55, 0x18, 0x68, 0xBB, 0x92, 0xA5, 0xA6, 0xB6, 0x03, 0xA5,
...
```

Replacing this file by a new version released by NXP leads to updating the FW of the connected PN7160 device.

Inside a nfc_example_FWupdate.c, there is #define FORCE_DWL. If set to true, it updates the FW every time (also if FW is already updated), if set to false, FW version on PN7160 is checked. If it is the same as in sFWupdate/phDnldNfc_UpdateSeq.c it skips FW update procedure, otherwise it proceeds into FW update procedure.

8 Abbreviations

Abbr.	Meaning
AN	Application Note
EMU	Emulation (card emulation)
GND	Ground
GPIO	General Purpose Input Output
HW	Hardware
I²C	Inter-Integrated Circuit (serial data bus)
IC	Integrated Circuit
IO	Input / Output
IRQ	Interrupt Request
NDA	Non Disclosure Agreement
NDEF	NFC Data Exchange Format
NFC	Near Field Communication
NFCC	NFC Controller
OS	Operating System
P2P	Peer to peer
PCD	Proximity Coupling Device (Contactless reader)
PIO	Programmed Input/Output
PICC	Proximity Integrated Circuit Card (Contactless card)
RF	Radiofrequency
RTOS	Real-Time Operating System
RST	Reset
R/W	Reader/Writer
SW	Software
T1T	Type 1 Tag (NFC Forum tag types definition)
T2T	Type 2 Tag (NFC Forum tag types definition)
T3T	Type 3 Tag (NFC Forum tag types definition)
T4T	Type 4 Tag (NFC Forum tag types definition)
T5T	Type 5 Tag (NFC Forum tag types definition)
VEN	V ENable pin (NFCC Hard reset control)

9 Legal information

9.1 Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

9.2 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.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

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.

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

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. - NXP B.V. is not an operating company and it does not distribute or sell products.

9.3 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.

9.4 Trademarks

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

NXP — wordmark and logo are trademarks of NXP B.V.

AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile — are trademarks and/or registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved.

DESFire — is a trademark of NXP B.V.

EdgeVerse — is a trademark of NXP B.V.

I2C-bus — logo is a trademark of NXP B.V.

MIFARE — is a trademark of NXP B.V.

MIFARE Classic — is a trademark of NXP B.V.

Tables

Figures

Contents

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

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

Date of release: 2 May 2023
Document identifier: AN13288