AN14224

How to use PN722x in Dual-Host mode

Rev. 3.0 — 7 February 2025

Application note

Document information

Information	Content
Keywords	PN722x, NCI, EMVCo, NFC Forum, Android, NFC, Dual-Host mode
Abstract	This document describes how to use PN722xBP2 with examples for the Kinetis K82 as the Secure MCU.

1 Introduction

This document describes how to use the PN722x in Dual-Host mode. Before getting started with operation in Dual-Host mode, it is important to gain basic knowledge about the PN722x (see [1]). The following sections describe the basic PN722x Dual-Host architecture, provide step-by-step instructions on how to prepare the MCUXpresso environment, and explain how to import and run prebuild examples available on the PN722x webpage (see [2]).

To follow the instructions in this document, the PNEV7220BP2 evaluation board is required. Any custom board can require different procedure (different Secure MCU, different protocol between Android host and Secure MCU, etc.).

2 Hardware

Dual-Host functionality can only be used with the PNEV722xBP2. The PNEV722xBP2 must be connected to the Android host.

The Kinetis K82, which is used as a Secure MCU, is part of the PNEV722xBP2 board. The K82 can be flashed with an external debugger like J-Link or any similar tool. The external debugger must be connected to the J35.

For detailed information on connections and where to connect the external debugger, refer to [1], specifically the section related to the PNEV722xBP2 HW.

3 Architecture

The PN722x Dual-Host architecture consists of three parts:

1. Android host
2. PN722x (part of the PNEV722xBP2)
3. Secure MCU (as part of the PNEV722xBP2 board: Kinetis K82 [3])

Figure 1 shows the PN722x Dual-Host architecture.

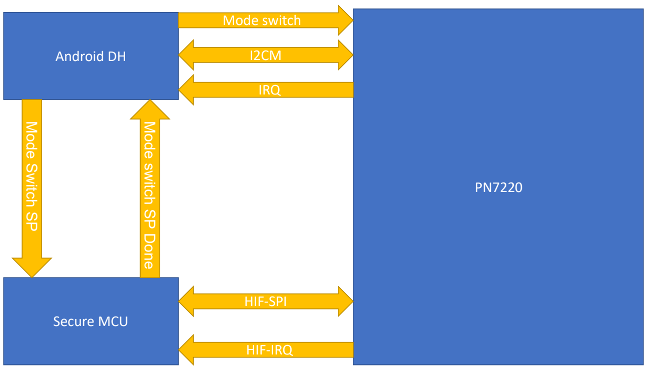


Figure 1. PN722x Dual-Host architecture

The Android host is the main device host (DH) andis responsible for driving the NFC Forum communication. It is connected to the PN722x with HIF2-I2C interface. Using Dual-Host architecture provides strict separation between operating in NFC Forum mode and EMVCo mode. The NFC Forum mode is executed on the Android DH, EMVCo mode is executed on the Secure MCU. The Secure MCU functions as the second DH and uses the HIF-SPI interface to communicate with the PN722x.

In this architecture, PN722x FW updates are only possible through the Secure MCU.

Note: *Standard PN722x FW updates through the Android DH as in Single-Host architecture are not possible in Dual-Host architecture.*

To switch between the Android DH and Secure MCU, the Mode Switch GPIO is used. By default, Mode Switch is set to low and the Android host communicates with the PN722x. When the Mode Switch GPIO is set to high,the PN722x is reset using the commands `CORE_RESET_CMD` and `CORE_INIT_CMD`. This reset is mandatory to clear all buffers and to prevent any unwanted data transfer between the Secure MCU and Android host. The Secure MCU is now responsible for communication with the PN722x.

The IRQ pin is used to inform the host when the PN722x wants to send responses to the DH. The IRQ line is the same for both the Android DH and Secure MCU.

But the connections mentioned above are not enough. As no library or stack support between Android host and Secure MCU is provided by NXP, a handshake mechanism is required. This handshake is performed with two GPIOs (see Table 1).

Table 1. Signal overview for handshake in Dual-Host architecture

GPIO	Android host	Secure MCU
Mode Switch SP	output	input
Mode Switch SP Done	input	output

Note: *Mode Switch SP Done is used only in case of the FW update example.*

The connections mentioned above are only examples. Users can define any protocol between Android host and Secure MCU.

While using PN722x in Dual-Host mode, the HIF2-I2C is connected to the Android DH. Unlike HIF1-I2C in Single-Host mode system which can act as wake-up source to PN722x, HIF2-I2C cannot wake-up PN722X from stand-by. In Dual-Host mode, the I2C2_WAKEUP pin will act as a wake-up source which needs to be used with the HIF2-I2C. By the connecting I2C2_WAKEUP pin to one of the HIF2-I2C pins we can wake-up PN722X due to any activity on HIF2-I2C pins.

4 Environment preparation

To set up the environment, the following resources must be downloaded:

- MCUXpresso Integrated Development Environment (IDE) [4]
- FRDM-K82 SDK [5]
- NCIRdLib examples [6]

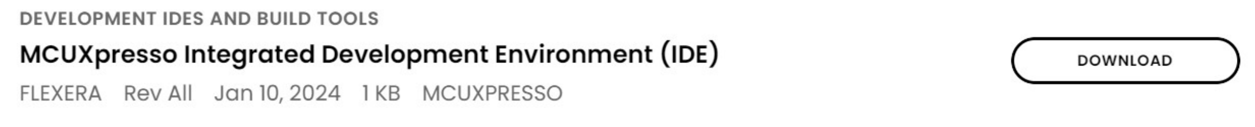
For instructions on how to import source code examples into the MCUXpresso, refer to the following subsections.

If only prebuild examples will be used, then a tool like J-Flash is needed to flash the Kinetis K82 directly with a debugger (J-Link, MCU-Link) connected to J35 on the PNEV722xBP2 board.

4.1 MCUXpresso installation

To download the necessary files for the installation:

1. Go to [4]
2. Scroll down to the section shown in Figure 2 and click the "DOWNLOAD" button
   Note: *The image below serves an example. The presentation on the webpage can change over time.*



DEVELOPMENT IDES AND BUILD TOOLS

MCUXpresso Integrated Development Environment (IDE)

FLEXERA Rev All Jan 10, 2024 1 KB MCUXPRESSO

Figure 2. MCUXpresso download

3. Log in to your NXP account to access the installation files.
4. After the successful login, select the desired version of the MCUXpresso and download it.
5. Run the installer and follow the instructions in Section 4.2.

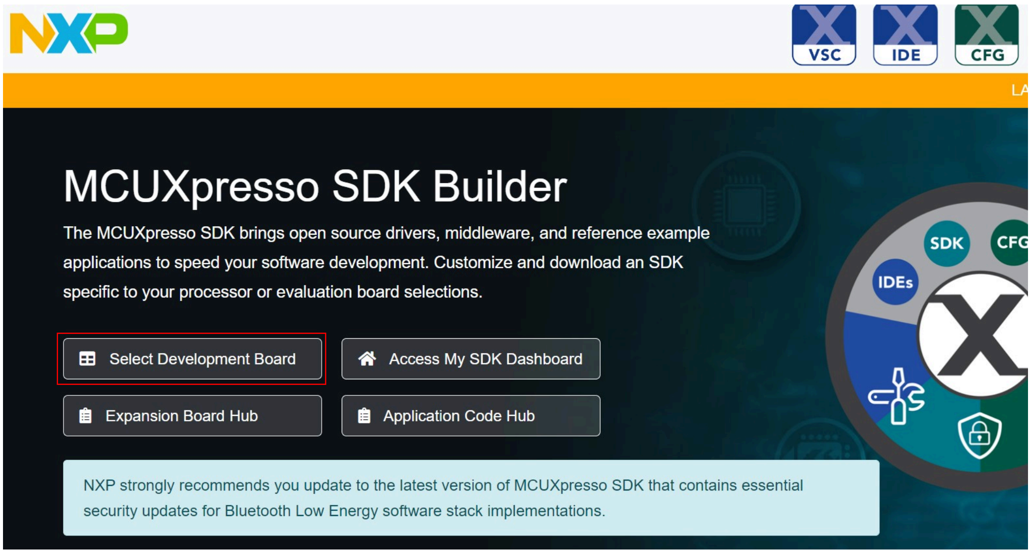
Additional information can be found in the following material:

- MCUXpresso IDE – user guide [7]
- MCUXpresso IDE – installation guide [8]

4.2 SDK installation

The PNEV722xBP2 board uses the FRDM-K82 as the Secure MCU. As all examples are based on K82, the FRDM-K82F SDK is needed to run the examples. The following instructions show how to acquire the SDK resources and install it in the MCUXpresso IDE.

1. Go to the MCUXpresso SDK Builder Webpage and select "Select Development Board".



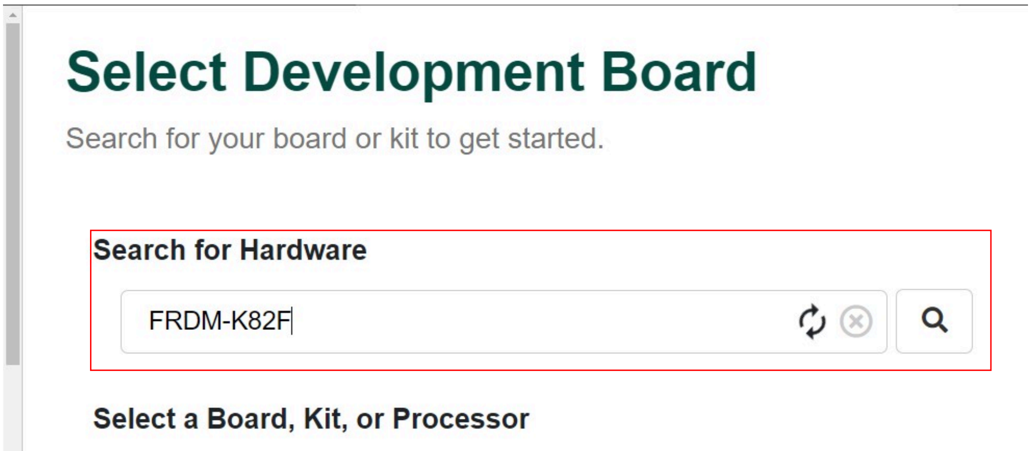
MCUXpresso SDK Builder

The MCUXpresso SDK brings open source drivers, middleware, and reference example applications to speed your software development. Customize and download an SDK specific to your processor or evaluation board selections.

NXP strongly recommends you update to the latest version of MCUXpresso SDK that contains essential security updates for Bluetooth Low Energy software stack implementations.

Figure 3. MCUXpresso SDK Builder webpage

2. Search for "FRDM-K82F".



Select Development Board

Search for your board or kit to get started.

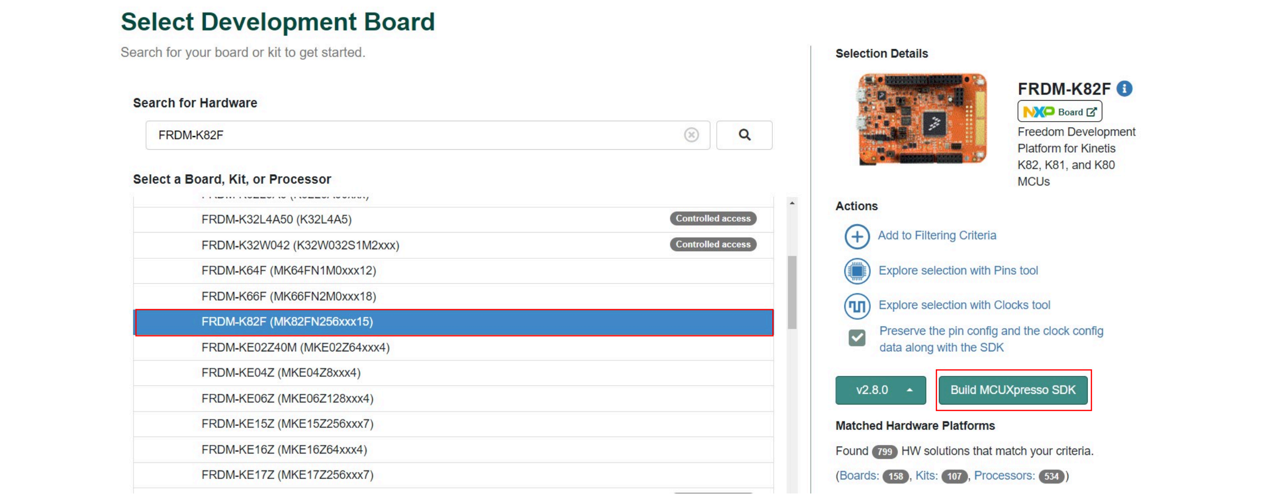
Search for Hardware

FRDM-K82F

Select a Board, Kit, or Processor

Figure 4. Search for specific HW

3. Select "FRDM-K82F" and click "Build MCUXpresso SDK".



Select Development Board

Search for your board or kit to get started.

Search for Hardware

FRDM-K82F

Select a Board, Kit, or Processor

- FRDM-K32L4A50 (K32L4A5) — Controlled access
- FRDM-K32W042 (K32W032S1M2xxx) — Controlled access
- FRDM-K64F (MK64FN1M0xxx12)
- FRDM-K66F (MK66FN2M0xxx18)
- FRDM-K82F (MK82FN256xxx15)
- FRDM-KE02Z40M (MKE02Z64xxx4)
- FRDM-KE04Z (MKE04Z8xxx4)
- FRDM-KE06Z (MKE06Z128xxx4)
- FRDM-KE15Z (MKE15Z256xxx7)
- FRDM-KE16Z (MKE16Z64xxx4)
- FRDM-KE17Z (MKE17Z256xxx7)

Selection Details

FRDM-K82F

NXP Board

Freedom Development Platform for Kinetis K82, K81, and K80 MCUs

Actions

- Add to Filtering Criteria
- Explore selection with Pins tool
- Explore selection with Clocks tool
- ☑ Preserve the pin config and the clock config data along with the SDK

v2.8.0 | Build MCUXpresso SDK

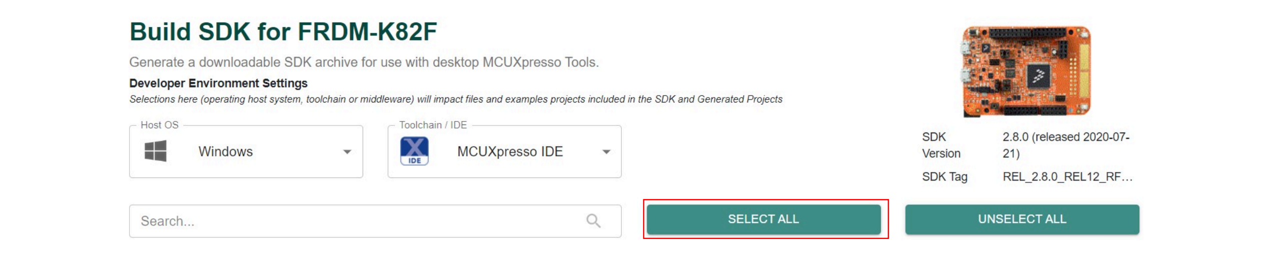
Matched Hardware Platforms

Found 799 HW solutions that match your criteria.

(Boards: 158, Kits: 107, Processors: 534)

Figure 5. Build the SDK

4. Select the necessary resources for the build.



Build SDK for FRDM-K82F

Generate a downloadable SDK archive for use with desktop MCUXpresso Tools.

Developer Environment Settings

Selections here (operating host system, toolchain or middleware) will impact files and examples projects included in the SDK and Generated Projects

Host OS: Windows

Toolchain / IDE: MCUXpresso IDE

SDK Version: 2.8.0 (released 2020-07-21)

SDK Tag: REL_2.8.0_REL12_RF…

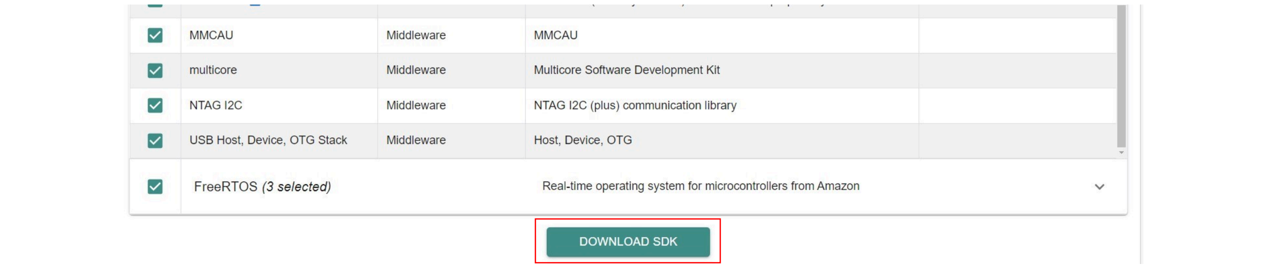
Search...

SELECT ALL | UNSELECT ALL

Figure 6. Select what must be added into the SDK

Note: *It is recommended to use the option "SELECT ALL".*

5. Click "DOWNLOAD SDK".



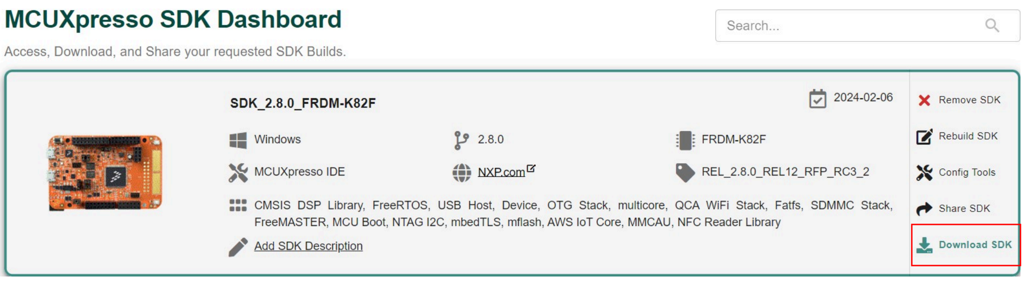
☑	MMCAU	Middleware	MMCAU
☑	multicore	Middleware	Multicore Software Development Kit
☑	NTAG I2C	Middleware	NTAG I2C (plus) communication library
☑	USB Host, Device, OTG Stack	Middleware	Host, Device, OTG

☑ FreeRTOS *(3 selected)* — Real-time operating system for microcontrollers from Amazon

DOWNLOAD SDK

Figure 7. Download all

6. Check the requested SDK build and click "Download SDK".



MCUXpresso SDK Dashboard

Access, Download, and Share your requested SDK Builds.

SDK_2.8.0_FRDM-K82F

2024-02-06

Windows | 2.8.0 | FRDM-K82F

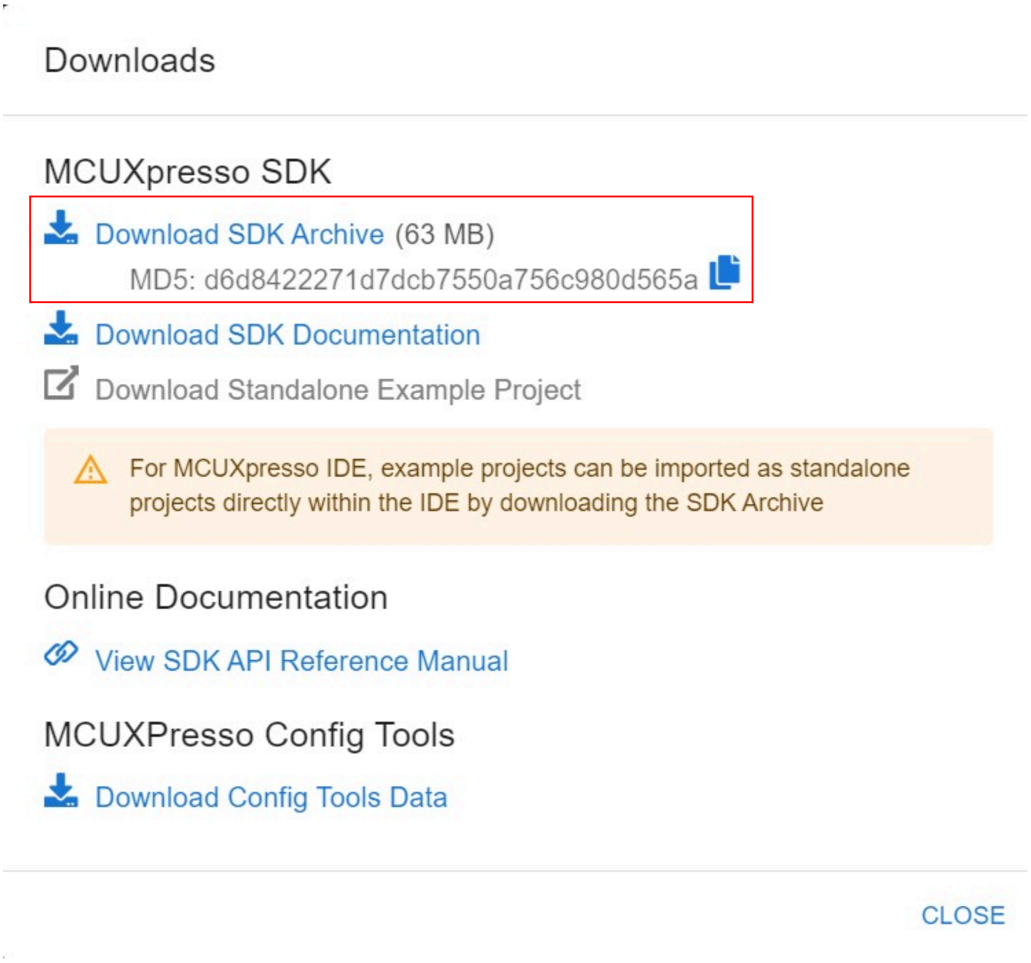
MCUXpresso IDE | NXP.com | REL_2.8.0_REL12_RFP_RC3_2

CMSIS DSP Library, FreeRTOS, USB Host, Device, OTG Stack, multicore, QCA WiFi Stack, Fatfs, SDMMC Stack, FreeMASTER, MCU Boot, NTAG I2C, mbedTLS, mflash, AWS IoT Core, MMCAU, NFC Reader Library

Add SDK Description

Figure 8. Confirm download

7. A window pops up. Click to "Download SDK Archive".



Downloads

MCUXpresso SDK

Download SDK Archive (63 MB)

MD5: d6d8422271d7dcb7550a756c980d565a

Download SDK Documentation

Download Standalone Example Project

For MCUXpresso IDE, example projects can be imported as standalone projects directly within the IDE by downloading the SDK Archive

Online Documentation

View SDK API Reference Manual

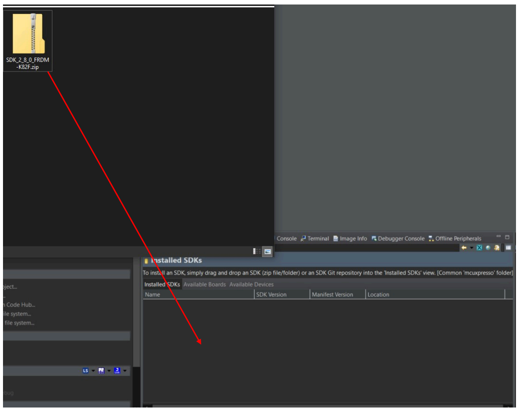
MCUXPresso Config Tools

Download Config Tools Data

Figure 9. Download SDK Archive

The next step is to import the SDK into the MCUXpresso IDE.

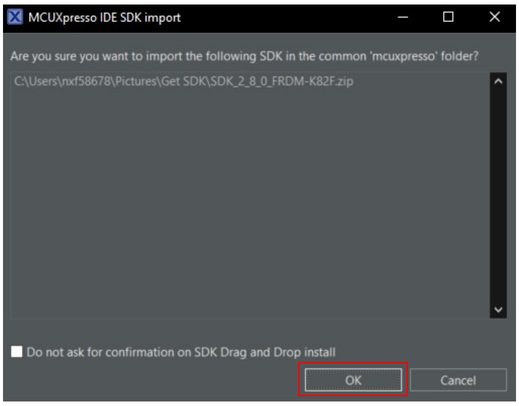
1. Drag the archived file into the MCUXpresso "Installed SDK's" section.



To install an SDK, simply drag and drop an SDK (zip file/folder) or an SDK Git repository into the 'Installed SDKs' view. [Common 'mcuxpresso' folder]

Figure 10. Drag&Drop SDK to MCUXpresso

2. Confirm by clicking the "OK" button.



Are you sure you want to import the following SDK in the common 'mcuxpresso' folder?

C:\Users\nxf58678\Pictures\Get SDK\SDK_2_8_0_FRDM-K82F.zip

□ Do not ask for confirmation on SDK Drag and Drop install

Figure 11. Click the "OK" button

This is only one way of installing the SDK. Refer to Importing and SDK package into MCUXpresso IDE for other options.

4.3 Import examples into MCUXpresso

Source code examples provided by NXP can be found in [6]. In the "*Design Resources*" section and "*Software*" subsection, download "*NciLib_PUB*". After downloading, unzip the package.

Follow the steps below to import them into the MCUXpresso IDE:

1. Click "Import project(s) from file system...".

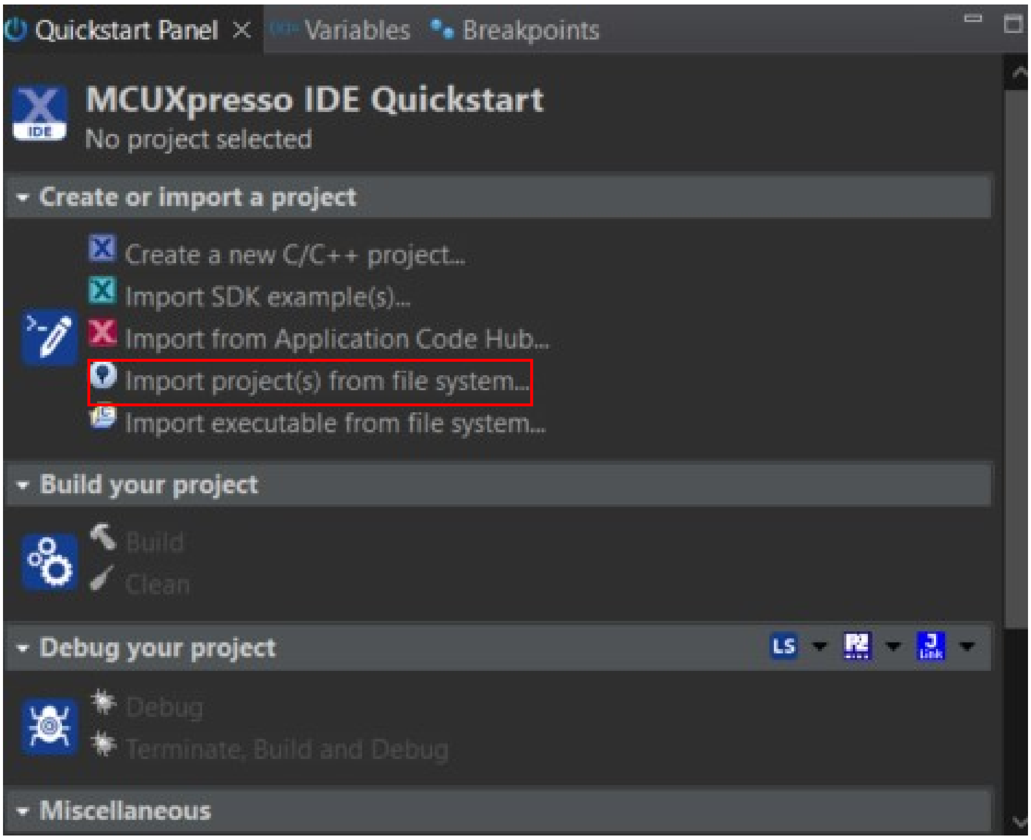
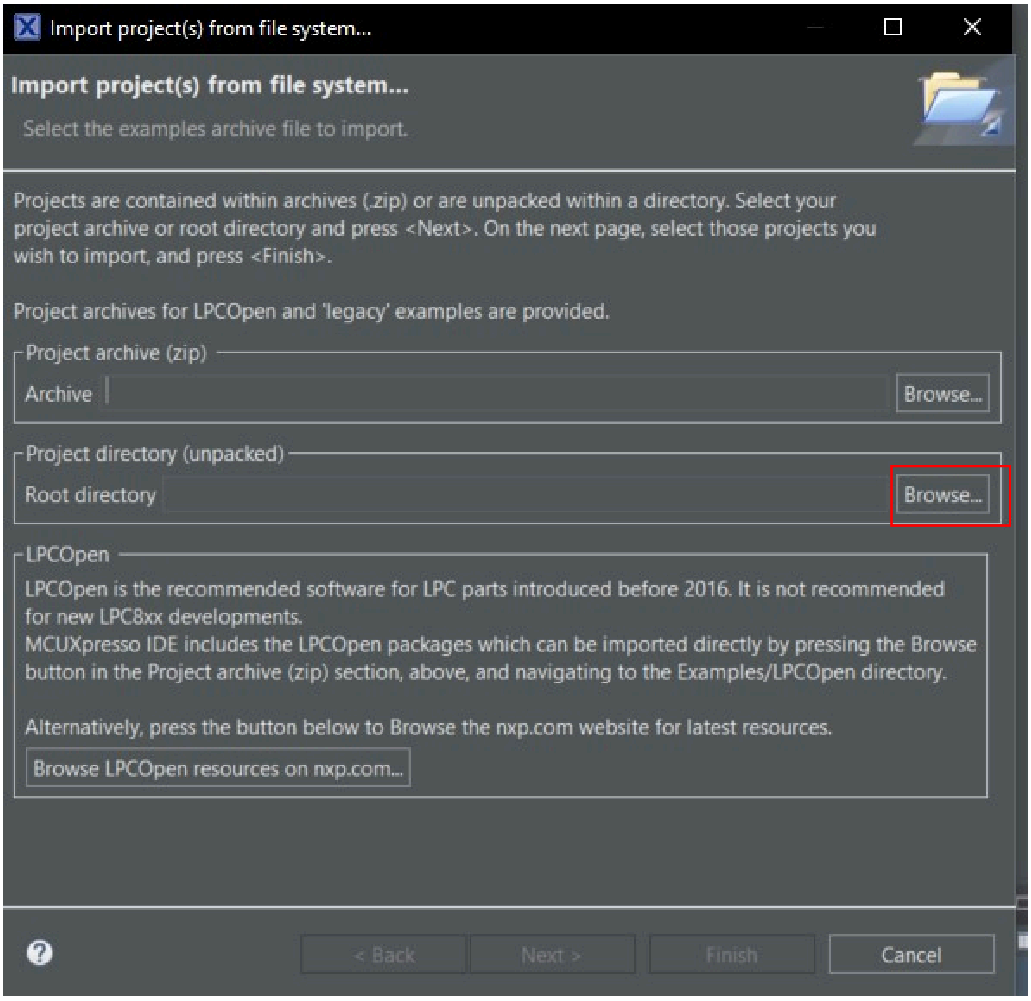

Quickstart Panel
Variables
Breakpoints
MCUXpresso IDE Quickstart
No project selected
Create or import a project
Create a new C/C++ project...
Import SDK example(s)...
Import from Application Code Hub...
Import project(s) from file system...
Import executable from file system...
Build your project
Build
Clean
Debug your project
Debug
Terminate, Build and Debug
Miscellaneous

Figure 12. Import project(s) from file system

2. Click "Browse" in "Project directory (unpacked)".



Import project(s) from file system...

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

Project directory (unpacked)

Root directory

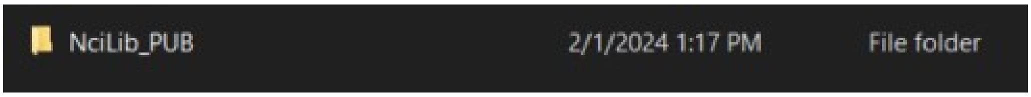
LPCOpen

LPCOpen is the recommended software for LPC parts introduced before 2016. It is not recommended for new LPC8xx developments.
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.

Figure 13. Browse for examples

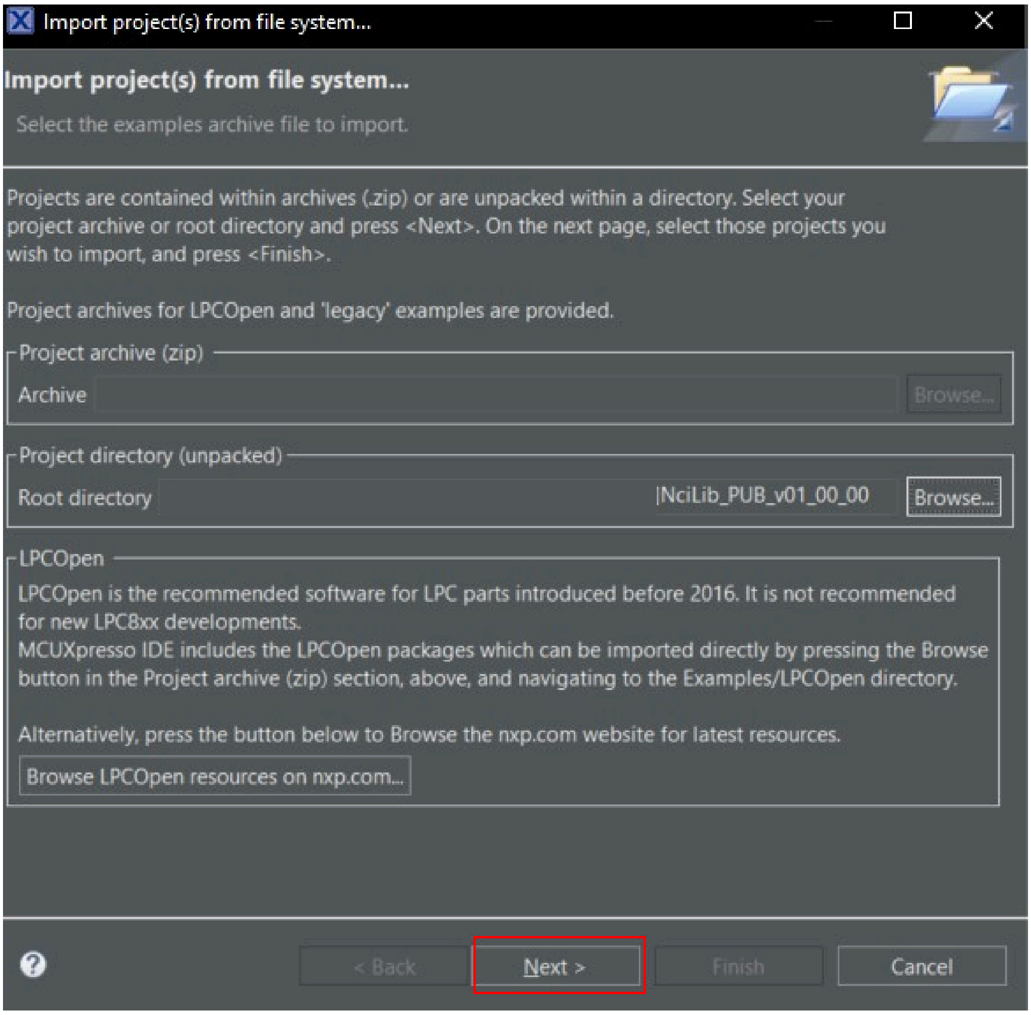
3. Search for the unzipped directory and select it.



NciLib_PUB 2/1/2024 1:17 PM File folder

Figure 14. Select an unzipped directory

4. Click "Next >".



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

Project directory (unpacked)

Root directory: |NciLib_PUB_v01_00_00

LPCOpen

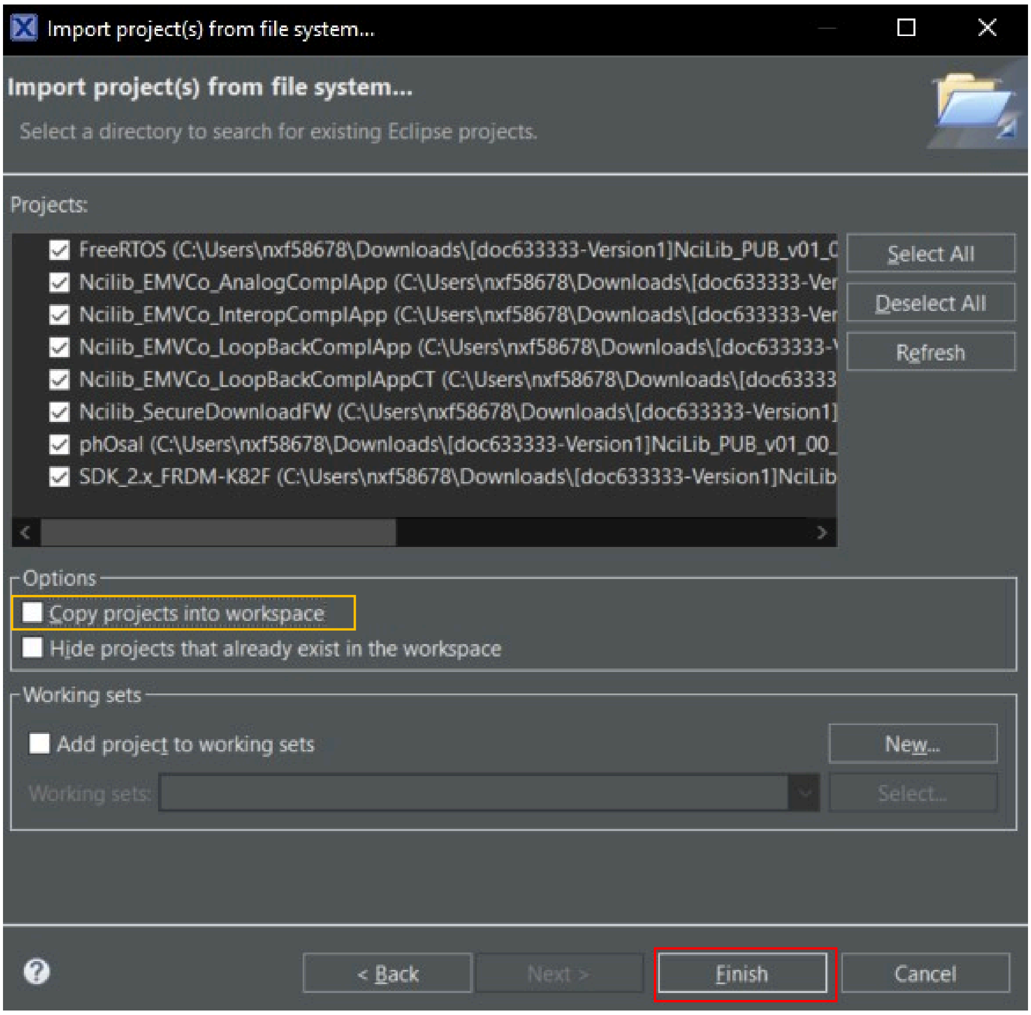
LPCOpen is the recommended software for LPC parts introduced before 2016. It is not recommended for new LPC8xx developments.
MCUXpresso IDE includes the LPCOpen packages which can be imported directly by pressing the Browse button in the Project archive (zip) section, above, and navigating to the Examples/LPCOpen 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 15. Click "Next >"

5. Deselect "Copy projects into workspace" and click "Finish".



Import project(s) from file system...

Select a directory to search for existing Eclipse projects.

Projects:

- ☑ FreeRTOS (C:\Users\nxf58678\Downloads\[doc633333-Version1]NciLib_PUB_v01_0
- ☑ Ncilib_EMVCo_AnalogComplApp (C:\Users\nxf58678\Downloads\[doc633333-Ver
- ☑ Ncilib_EMVCo_InteropComplApp (C:\Users\nxf58678\Downloads\[doc633333-Ver
- ☑ Ncilib_EMVCo_LoopBackComplApp (C:\Users\nxf58678\Downloads\[doc633333-V
- ☑ Ncilib_EMVCo_LoopBackComplAppCT (C:\Users\nxf58678\Downloads\[doc63333
- ☑ Ncilib_SecureDownloadFW (C:\Users\nxf58678\Downloads\[doc633333-Version1]
- ☑ phOsal (C:\Users\nxf58678\Downloads\[doc633333-Version1]NciLib_PUB_v01_00_
- ☑ SDK_2.x_FRDM-K82F (C:\Users\nxf58678\Downloads\[doc633333-Version1]NciLib

Options

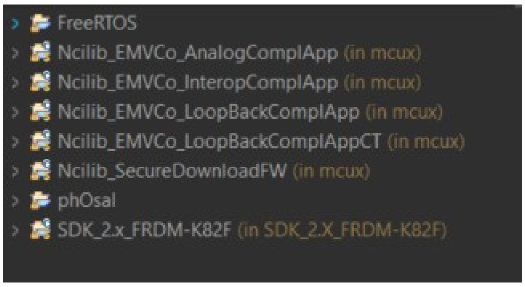
- □ Copy projects into workspace
- □ Hide projects that already exist in the workspace

Working sets

- □ Add project to working sets

Figure 16. Unclick "Copy project into workspace" and click "Finish"

6. Examples are now imported and ready to use.



FreeRTOS
Ncilib_EMVCo_AnalogComplApp (in mcux)
Ncilib_EMVCo_InteropComplApp (in mcux)
Ncilib_EMVCo_LoopBackComplApp (in mcux)
Ncilib_EMVCo_LoopBackComplAppCT (in mcux)
Ncilib_SecureDownloadFW (in mcux)
phOsal
SDK_2.x_FRDM-K82F (in SDK_2.X_FRDM-K82F)

Figure 17. Imported examples

All examples except Ncilib_SecureDownloadFW can be built in Debug and Release mode. Ncilib_SecureDownloadFW requires input from the PC (*.esfwu* file) and can therefore be built only in Debug mode.

5 Explanation of applications

- **Secure MCU mode switch application**: Installation instructions can be found in [1]. This application is responsible for toggling the Mode switch pin. It is an important application, since all examples in NCIRdLib are waiting for the mode switch to toggle. This application exists on the Android host.
- **Ncilib_EMVCo_AnalogComplApp**: This application is used to perform EMVCo3.0(L1) Analog compliance validation.
- **Ncilib_EMVCo_InteropComplApp:** This example is an interoperability loopback application, which is used to perform EMVCo IOP(L1) with add-on (TTA Bulletin No.195) compliance validation.
- **Ncilib_EMVCo_LoopBackComplApp:** For the EMVCo profile, this example provides a full EMVCo digital demonstration along with to SELECT PPSE Commands. This application is used to perform EMVCo CLIF compliance validation.
- **Ncilib_EMVCo_LoopBackComplAppCT:** This application is used to perform EMVCo CT compliance validation. This application will automatically open and close the TDA connection as soon as a contact card is inserted or removed from the TDA slot.
- **Ncilib_SecureDownloadFW:** This application is used to perform secure firmware download. For information on how to execute this example, refer to Section 6.1.

6 Running the examples

Users have two options to execute the examples:

1. Flash prebuild examples directly to K82
2. Use MCUXpresso to build the example and flash it directly to K82

This section describes how to run the examples. As mentioned earlier, the main host is the Android device. Therefore, it is necessary to use the "Secure MCU Mode Switch application" (refer to the PN7220 Quick start guide). With this application, the "Mode switch" pin toggles based on the selected setting. This is required to inform PN722x that Secure MCU is taking over the communication.

- **Option 1: Flash prebuild examples directly to K82**
  It is faster and MCUXpresso is not required. It can flash the K82 via J35 with tool like J-Flash. Prebuild examples are avaiable in [1] under "*Design resources*" section and "*Software*" subsection named as "*NciLib_PUB_Prebuild*". Since prebuild examples are built in Release mode, there are no output logs where user can see what is happening.
  The steps for using Secure MCU Mode Switch application are the same as in option 2. The GPIO toggle is also the same for both options.
  User can see what is happening on the board through the LEDs on the PNEV722xBP2 board:
  – D16 (yellow LED) is ON → NFC Forum mode is active
  – D11 (green LED) is ON → EMVCo mode is active

Figure 18 shows the location of the LEDs. More information on the LED can be found in schematics found on [2].

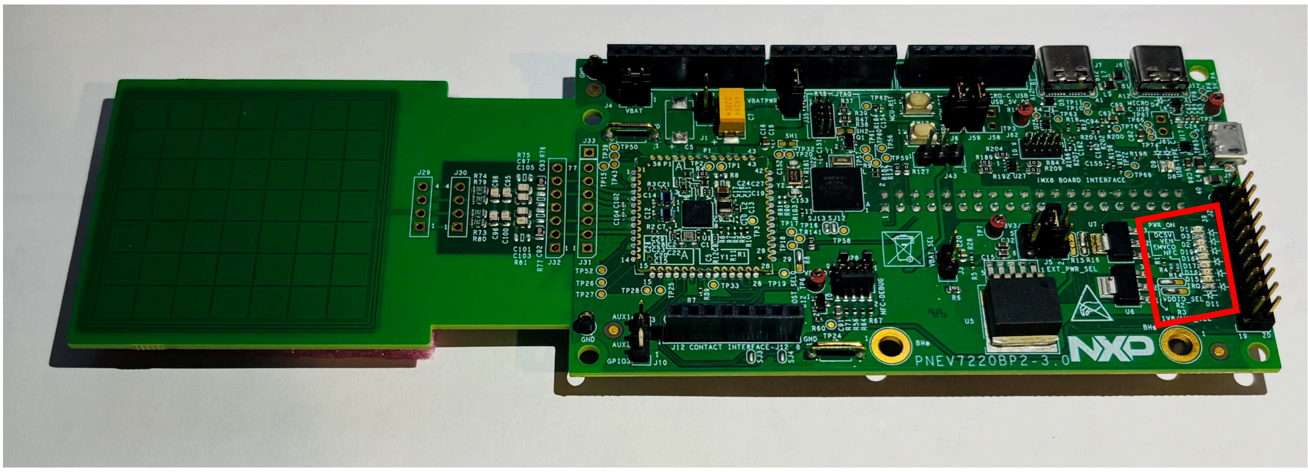


Figure 18. LED location on PNEV722xBP2 board

- **Option 2: Use MCUXpresso to build the example and flash it directly to K82**
  It is a bit slower in comparison to option 1 as setup of the environment and porting of the examples is required. However, the benefit is that users can adapt the code of the example to their needs and observe functionality through the available debug logs. Find step-by-step instruction on how to run examples with option 2 below.

The following sections show how to run an EMVCo loopback example:

1. Flash the K82 with *Ncilib_EMVCo_LoopBackComplApp*. To flash the K82, connect the debugger (J-Link, MCU-Link) to the J35 on the PNEV722xBP2 board.

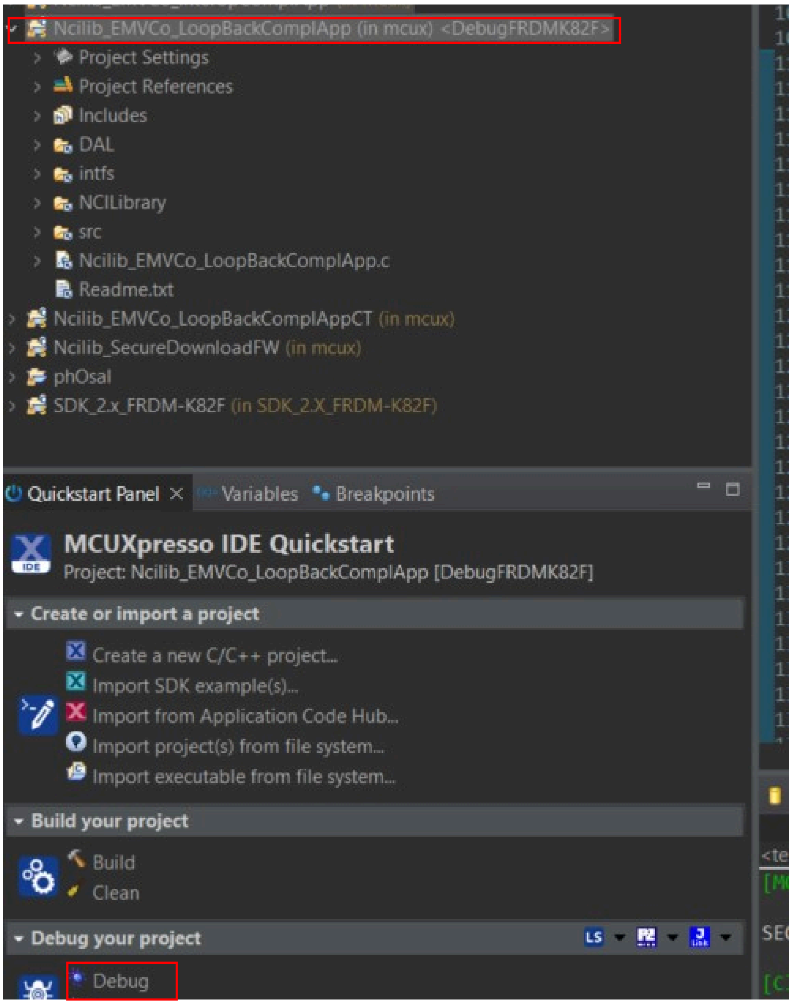
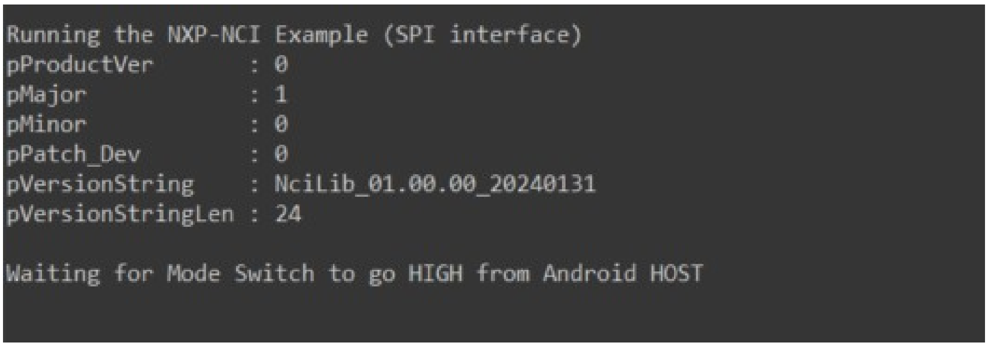

Ncilib_EMVCo_LoopBackComplApp (in mcux) <DebugFRDMK82F>
Project Settings
Project References
Includes
DAL
intfs
NCILibrary
src
Ncilib_EMVCo_LoopBackComplApp.c
Readme.txt
Ncilib_EMVCo_LoopBackComplAppCT (in mcux)
Ncilib_SecureDownloadFW (in mcux)
phOsal
SDK_2.x_FRDM-K82F (in SDK_2.X_FRDM-K82F)
Quickstart Panel
Variables
Breakpoints
MCUXpresso IDE Quickstart
Project: Ncilib_EMVCo_LoopBackComplApp [DebugFRDMK82F]
Create or import a project
Create a new C/C++ project...
Import SDK example(s)...
Import from Application Code Hub...
Import project(s) from file system...
Import executable from file system...
Build your project
Build
Clean
Debug your project
Debug

Figure 19. Run the example

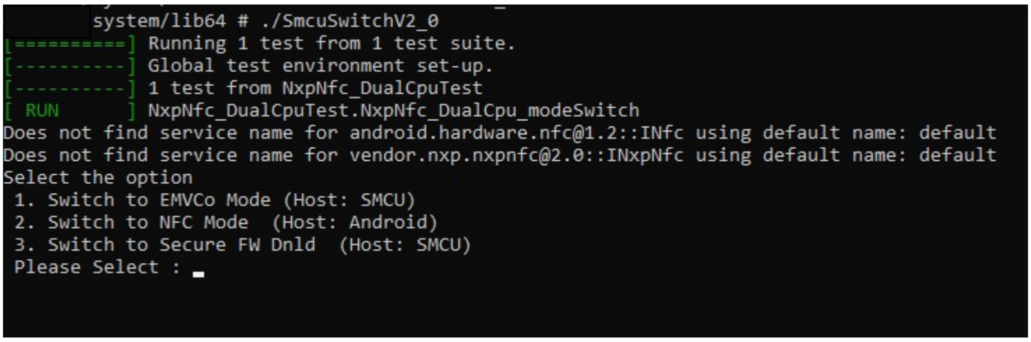
2. After running the example, it waits for the Mode switch to toggle high (the same for all applications).



```
Running the NXP-NCI Example (SPI interface)
pProductVer        : 0
pMajor             : 1
pMinor             : 0
pPatch_Dev         : 0
pVersionString     : NciLib_01.00.00_20240131
pVersionStringLen  : 24

Waiting for Mode Switch to go HIGH from Android HOST
```

Figure 20. Application wait Mode switch to toggle high

3. Now, run the Secure MCU Mode Switch application on tge Android host



```
[redacted] system/lib64 # ./SmcuSwitchV2_0
[==========] Running 1 test from 1 test suite.
[----------] Global test environment set-up.
[----------] 1 test from NxpNfc_DualCpuTest
[ RUN      ] NxpNfc_DualCpuTest.NxpNfc_DualCpu_modeSwitch
Does not find service name for android.hardware.nfc@1.2::INfc using default name: default
Does not find service name for vendor.nxp.nxpnfc@2.0::INxpNfc using default name: default
Select the option
 1. Switch to EMVCo Mode (Host: SMCU)
 2. Switch to NFC Mode  (Host: Android)
 3. Switch to Secure FW Dnld  (Host: SMCU)
 Please Select : _
```

Figure 21. Run Secure MCU Mode Switch application

- If 1 is selected, the Mode Switch pin goes high and the Secure MCU runs the EMVCo profile.
- If 2 is selected, the Mode Switch pin goes low and the Android host runs the NFC Forum profile.
- If 3 is selected, the Mode Switch pin goes high and Secure MCU starts with FW update.

4. Select 1 and press enter.
   - Initially, all three pins (MODE_SWITCH, MODE_SWITCH_SP and MODE_SWITCH_SP_DONE) are low.

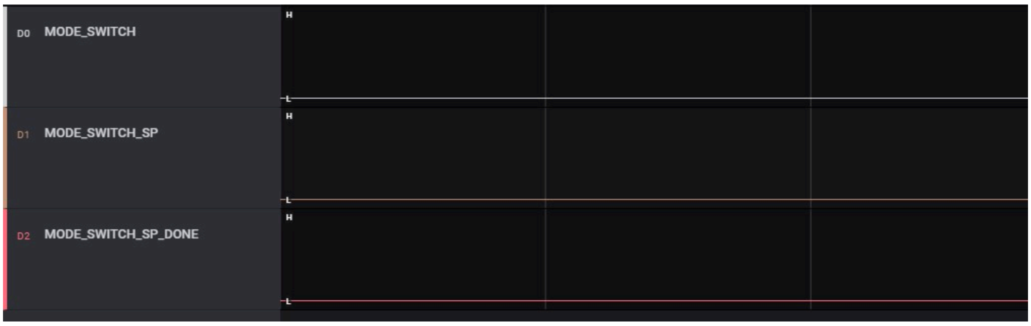


Figure 22. Initial values on pins

   - When 1 is selected, MODE_SWITCH and MODE_SWITCH_SP will toggle high.

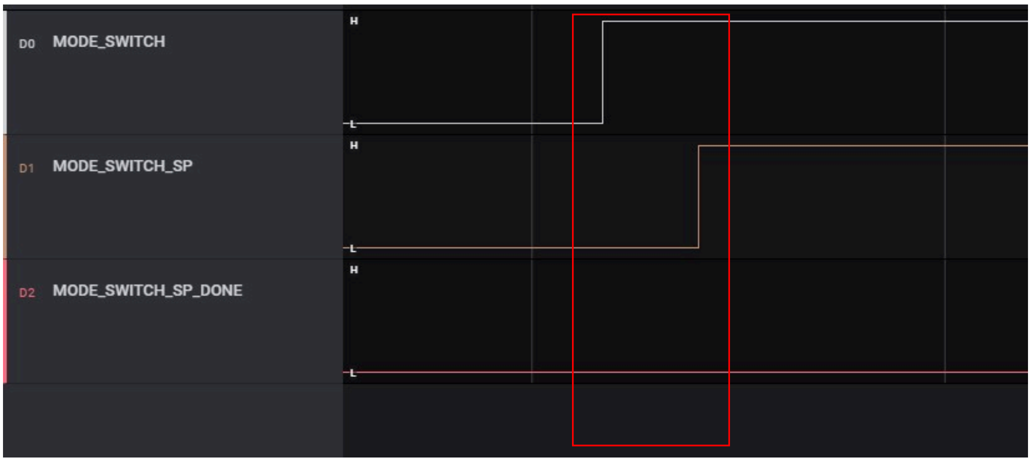
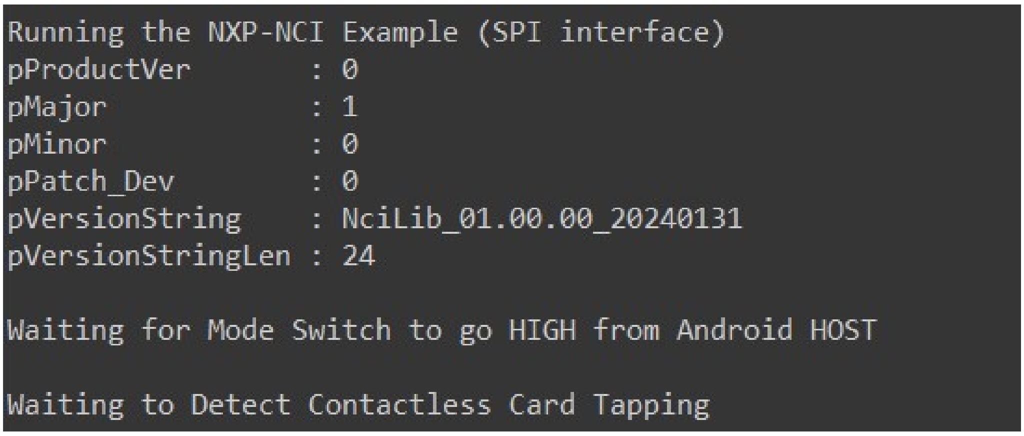


Figure 23. Pins change value to high

   - Now, the Secure MCU has taken over and PN722x is running in EMVCo mode. PN722x can now detect a card in EMVCo mode. There is communication between the PN722x and the Secure MCU.



```
Running the NXP-NCI Example (SPI interface)
pProductVer       : 0
pMajor            : 1
pMinor            : 0
pPatch_Dev        : 0
pVersionString    : NciLib_01.00.00_20240131
pVersionStringLen : 24

Waiting for Mode Switch to go HIGH from Android HOST

Waiting to Detect Contactless Card Tapping
```

Figure 24. PN722x can detect card in EMVCo mode

5. - If 2 is selected, the pins toggle to low and Android takes over the communication (NFC Forum).

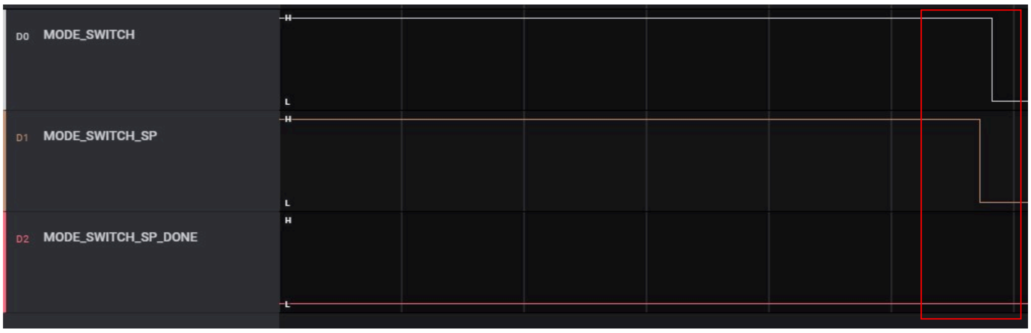


Figure 25. Changing back to Android host (NFC Forum)

Based on the choices in the Secure MCU Mode Switch application, the values of three pins change, and the selected host is responsible for communication.

The steps described above are common to all examples, except for the FW update, where a different option is enabled in "Secure MCU Mode Switch" (option 3). However, this does not affect the execution of the examples. To see how the FW update procedure can be executed, see Section 6.1.

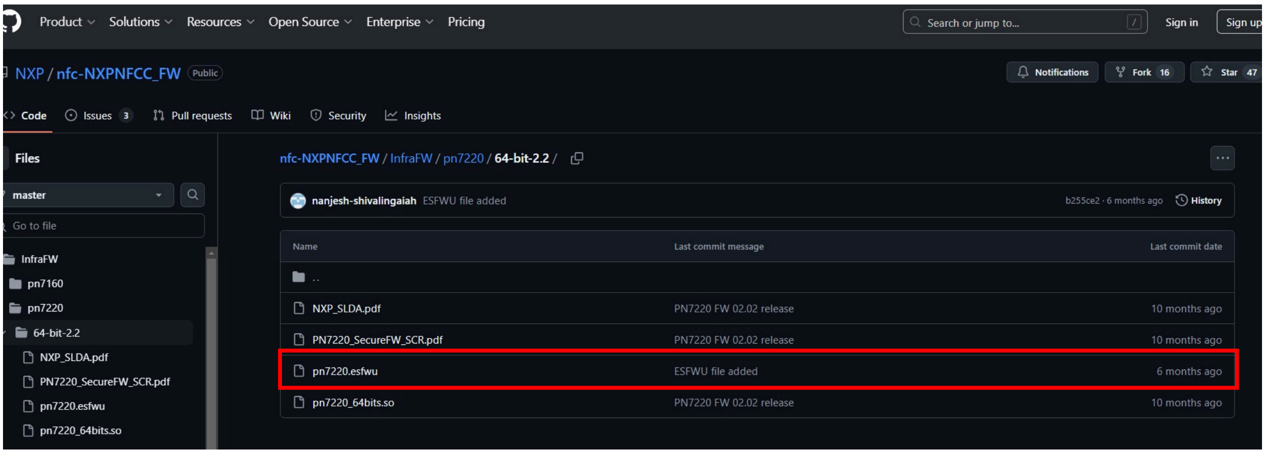
6.1 Secure FW update

As mentioned in Section 3, the only way to execute the FW update example in Dual-Host mode is to set up the environment (see Section 4) since the example can only be built in Debug mode.

This chapter explains the step-by-step procedure how to perform the FW update. The environment needs to be prepared and the source code examples need to be imported into the MCUXpresso before following the steps below.

To execute this example the *.esfwu* file is needed. To acquire the FW, follow the steps below:

1. Go to [10] and select the FW version and click the *.esfwu* file.



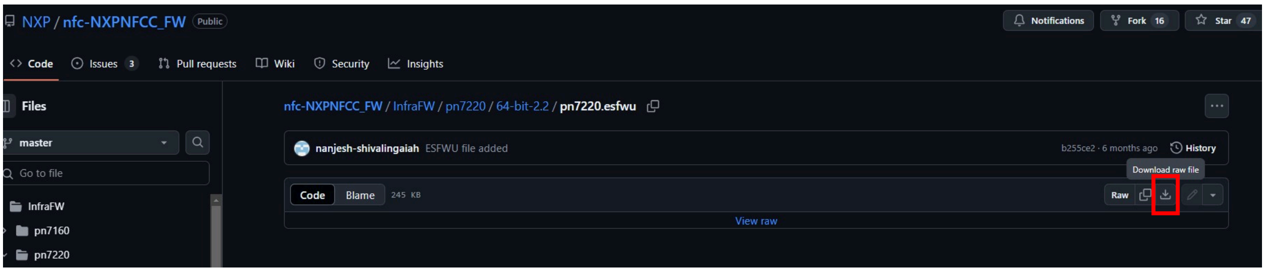
nfc-NXPNFCC_FW / InfraFW / pn7220 / 64-bit-2.2 /

nanjesh-shivalingaiah ESFWU file added b255ce2 · 6 months ago History

Name	Last commit message	Last commit date
..		
NXP_SLDA.pdf	PN7220 FW 02.02 release	10 months ago
PN7220_SecureFW_SCR.pdf	PN7220 FW 02.02 release	10 months ago
pn7220.esfwu	ESFWU file added	6 months ago
pn7220_64bits.so	PN7220 FW 02.02 release	10 months ago

Figure 26. Select FW file

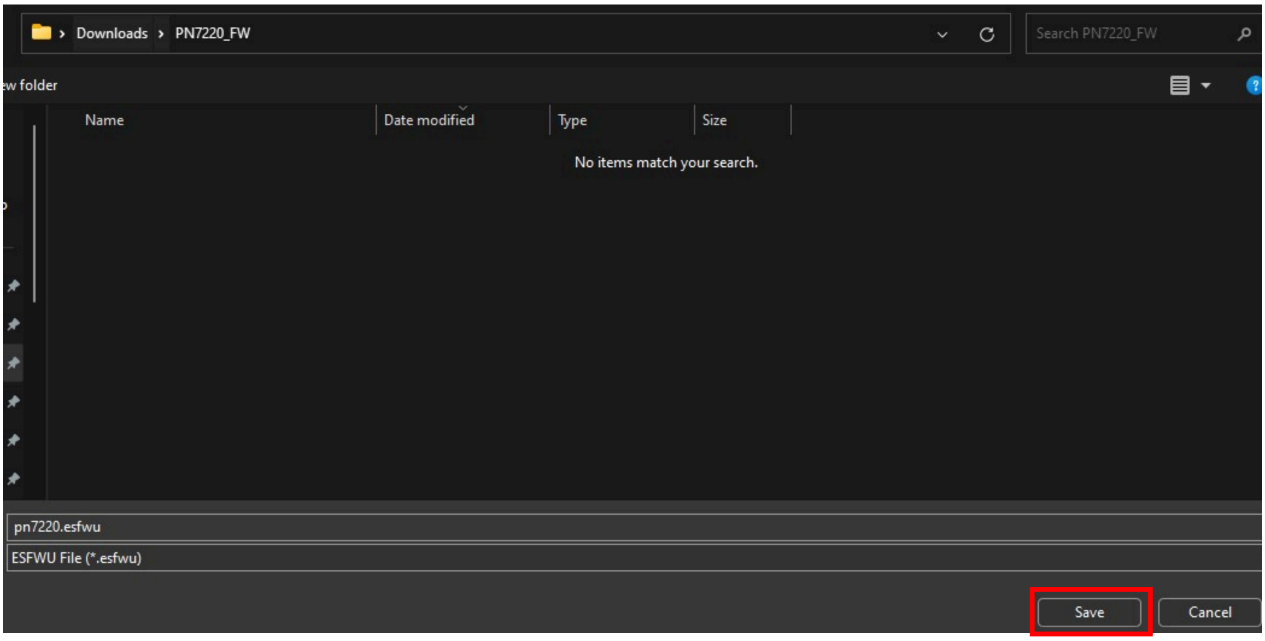
2. Click on **Download raw frame** and save the file.



nfc-NXPNFCC_FW / InfraFW / pn7220 / 64-bit-2.2 / pn7220.esfwu

nanjesh-shivalingaiah ESFWU file added b255ce2 · 6 months ago History

Figure 27. Click on download button

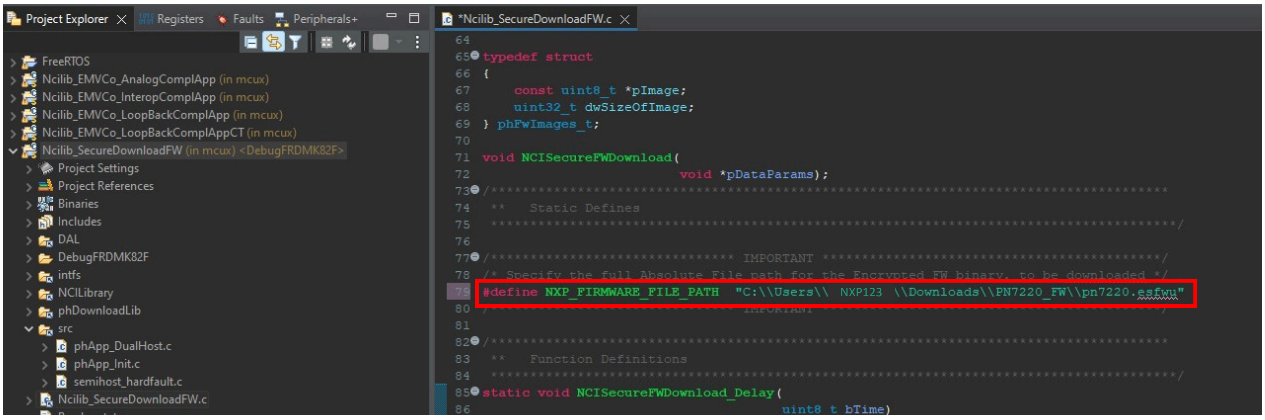


pn7220.esfwu

ESFWU File (*.esfwu)

Figure 28. Save file

3. In *Ncilib_SecureDownloadFW.c*, change the *NXP_FIRMWARE_FILE_PATH* variable to indicate the location of the *.esfwu* file.

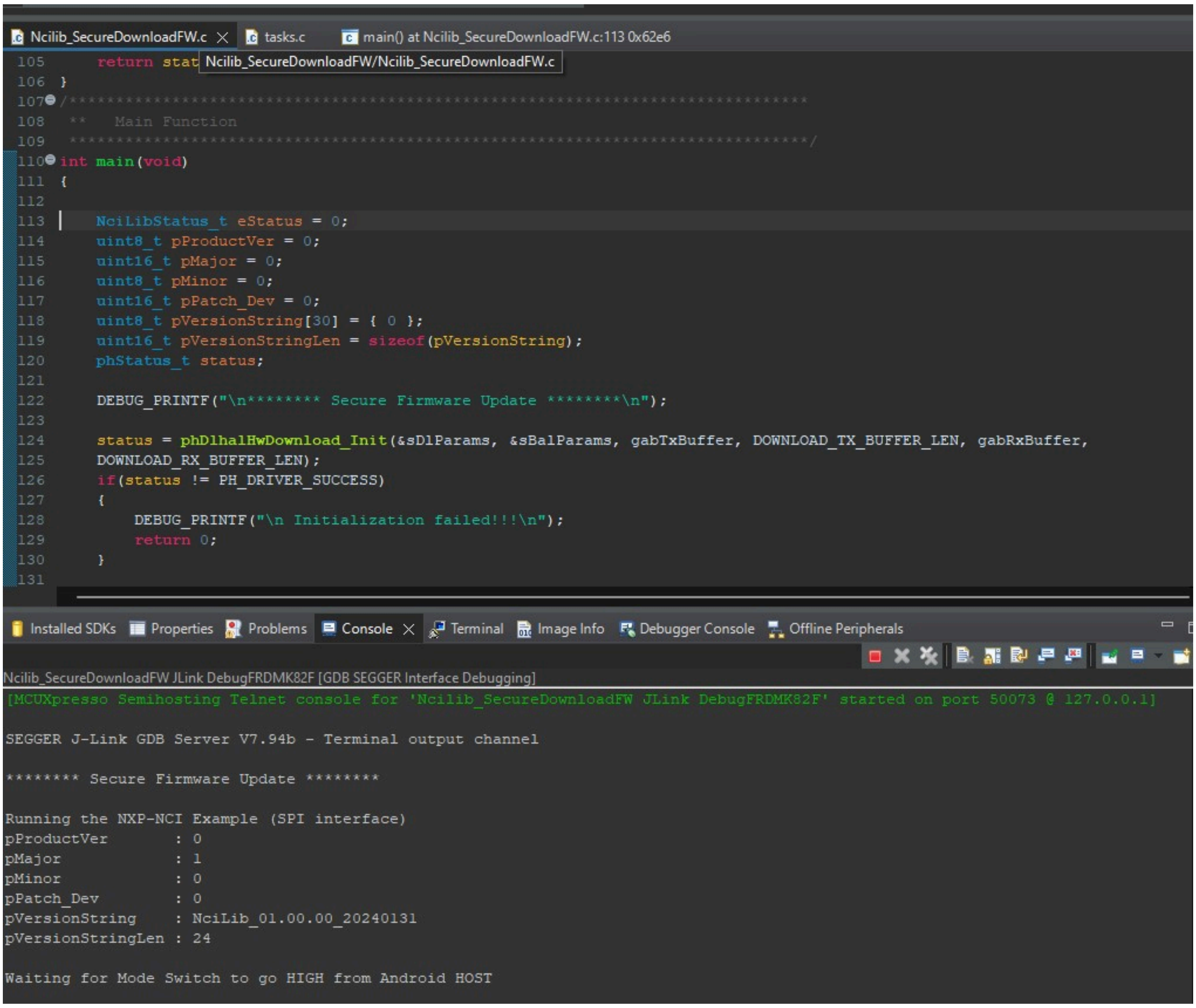


Project Explorer
Registers
Faults
Peripherals+
FreeRTOS
Ncilib_EMVCo_AnalogComplApp (in mcux)
Ncilib_EMVCo_InteropComplApp (in mcux)
Ncilib_EMVCo_LoopBackComplApp (in mcux)
Ncilib_EMVCo_LoopBackComplAppCT (in mcux)
Ncilib_SecureDownloadFW (in mcux) <DebugFRDMK82F>
Project Settings
Project References
Binaries
Includes
DAL
DebugFRDMK82F
intfs
NCILibrary
phDownloadLib
src
phApp_DualHost.c
phApp_Init.c
semihost_hardfault.c
Ncilib_SecureDownloadFW.c
*Ncilib_SecureDownloadFW.c

```

typedef struct
{
    const uint8_t *pImage;
    uint32_t dwSizeOfImage;
} phFwImages_t;

void NCISecureFWDownload(
                                void *pDataParams);
/******************************************************************************
 **   Static Defines
 ******************************************************************************/

/****************************** IMPORTANT *******************************************/
/* Specify the full Absolute File path for the Encrypted FW binary, to be downloaded */
#define NXP_FIRMWARE_FILE_PATH  "C:\\Users\\ NXP123 \\Downloads\\PN7220_FW\\pn7220.esfwu"
/****************************** IMPORTANT *******************************************/

/******************************************************************************
 **   Function Definitions
 ******************************************************************************/
static void NCISecureFWDownload_Delay(
                                        uint8_t bTime)
```

Figure 29. Change NXP_FIRMWARE_FILE_PATH to indicate the location of FW

4. Run the example.



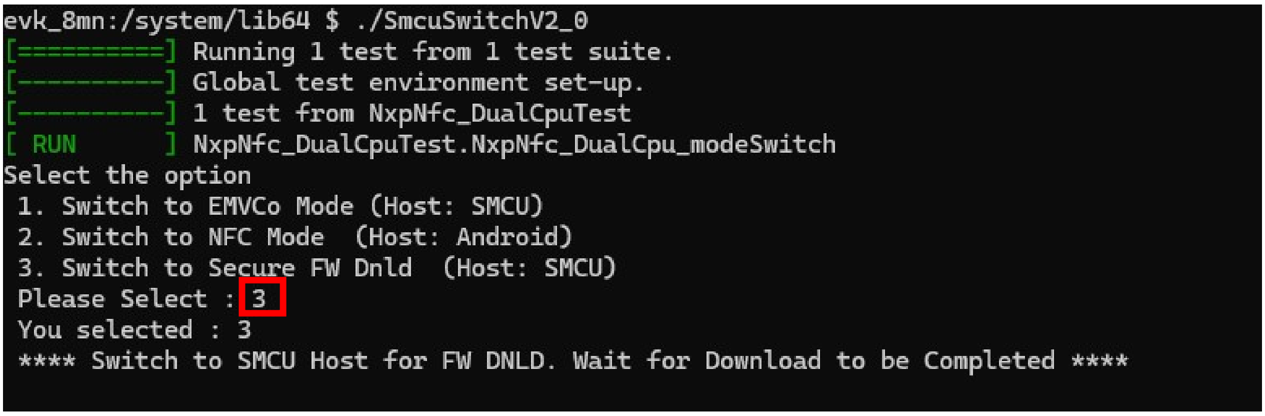
Ncilib_SecureDownloadFW.c
tasks.c
main() at Ncilib_SecureDownloadFW.c:113 0x62e6
Ncilib_SecureDownloadFW/Ncilib_SecureDownloadFW.c
Installed SDKs
Properties
Problems
Console
Terminal
Image Info
Debugger Console
Offline Peripherals

```
    return stat
}
/*********************************************************************************
**   Main Function
*********************************************************************************/
int main(void)
{

    NciLibStatus_t eStatus = 0;
    uint8_t pProductVer = 0;
    uint16_t pMajor = 0;
    uint8_t pMinor = 0;
    uint16_t pPatch_Dev = 0;
    uint8_t pVersionString[30] = { 0 };
    uint16_t pVersionStringLen = sizeof(pVersionString);
    phStatus_t status;

    DEBUG_PRINTF("\n******** Secure Firmware Update ********\n");

    status = phDlhalHwDownload_Init(&sDlParams, &sBalParams, gabTxBuffer, DOWNLOAD_TX_BUFFER_LEN, gabRxBuffer,
    DOWNLOAD_RX_BUFFER_LEN);
    if(status != PH_DRIVER_SUCCESS)
    {
        DEBUG_PRINTF("\n Initialization failed!!!\n");
        return 0;
    }

```

```
Ncilib_SecureDownloadFW JLink DebugFRDMK82F [GDB SEGGER Interface Debugging]
[MCUXpresso Semihosting Telnet console for 'Ncilib_SecureDownloadFW JLink DebugFRDMK82F' started on port 50073 @ 127.0.0.1]

SEGGER J-Link GDB Server V7.94b - Terminal output channel

******** Secure Firmware Update ********

Running the NXP-NCI Example (SPI interface)
pProductVer       : 0
pMajor            : 1
pMinor            : 0
pPatch_Dev        : 0
pVersionString    : NciLib_01.00.00_20240131
pVersionStringLen : 24

Waiting for Mode Switch to go HIGH from Android HOST
```

Figure 30. Run example

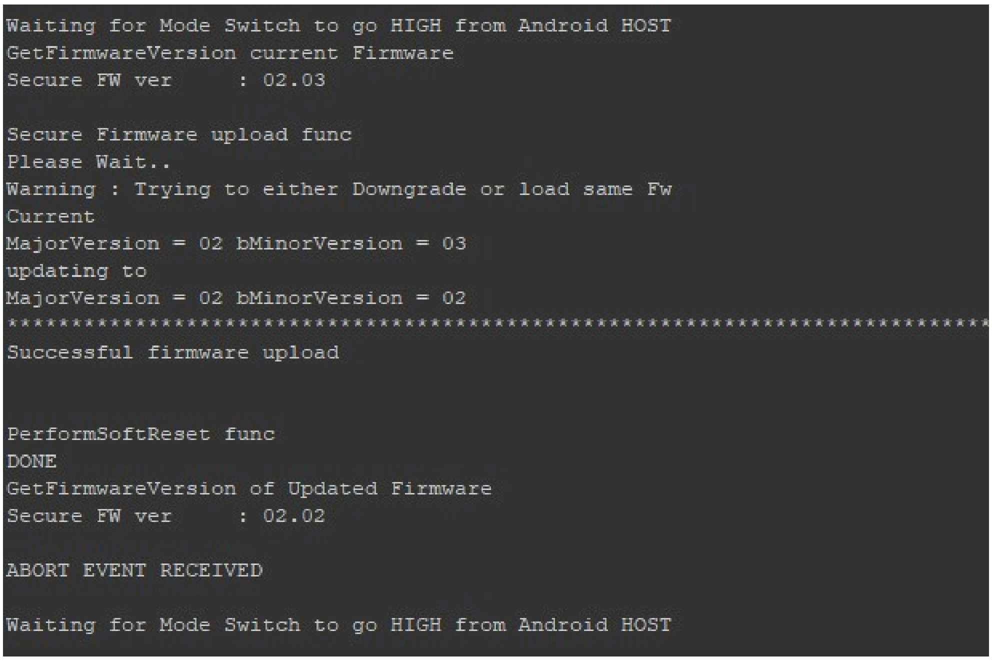
5. On the Android host, run the Secure MCU Mode Switch application and select option 3:



```
evk_8mn:/system/lib64 $ ./SmcuSwitchV2_0
[==========] Running 1 test from 1 test suite.
[----------] Global test environment set-up.
[----------] 1 test from NxpNfc_DualCpuTest
[ RUN      ] NxpNfc_DualCpuTest.NxpNfc_DualCpu_modeSwitch
Select the option
 1. Switch to EMVCo Mode (Host: SMCU)
 2. Switch to NFC Mode  (Host: Android)
 3. Switch to Secure FW Dnld  (Host: SMCU)
 Please Select : 3
 You selected : 3
 **** Switch to SMCU Host for FW DNLD. Wait for Download to be Completed ****
```

Figure 31. Secure MCU Mode Switch application

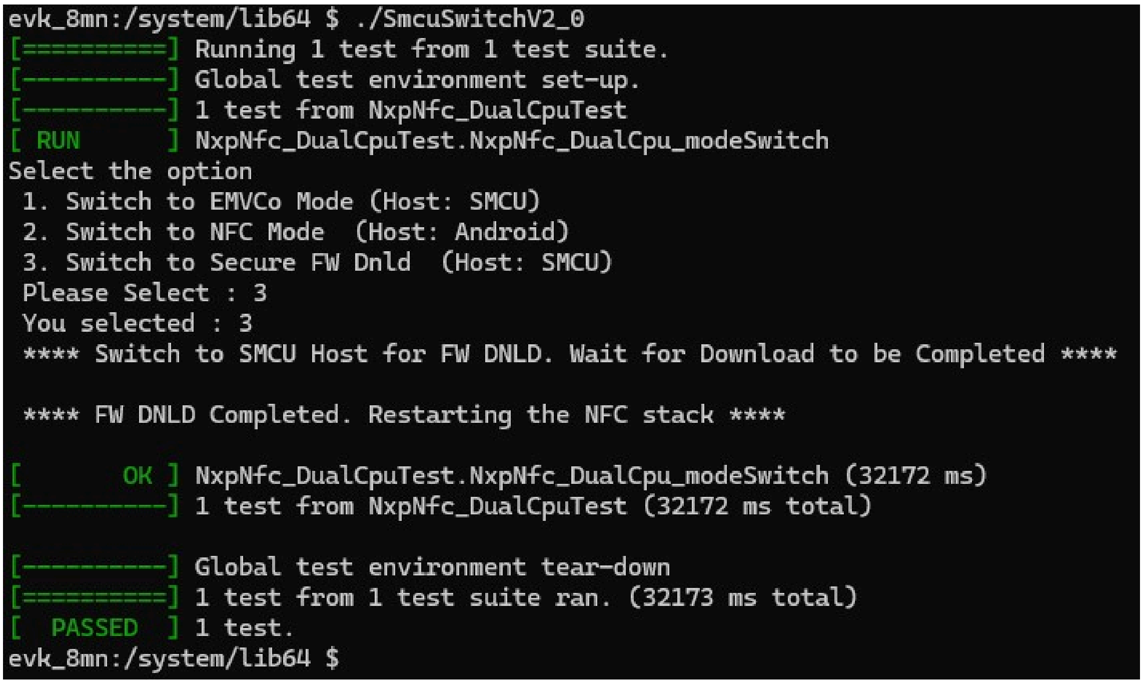
6. The FW update procedure will now start.



```
Waiting for Mode Switch to go HIGH from Android HOST
GetFirmwareVersion current Firmware
Secure FW ver     : 02.03

Secure Firmware upload func
Please Wait..
Warning : Trying to either Downgrade or load same Fw
Current
MajorVersion = 02 bMinorVersion = 03
updating to
MajorVersion = 02 bMinorVersion = 02
*****************************************************************************
Successful firmware upload


PerformSoftReset func
DONE
GetFirmwareVersion of Updated Firmware
Secure FW ver     : 02.02

ABORT EVENT RECEIVED

Waiting for Mode Switch to go HIGH from Android HOST
```

Figure 32. Output from MCUXpresso before/during/after FW update procedure

7. The Secure MCU Mode Switch application will finish:



```
evk_8mn:/system/lib64 $ ./SmcuSwitchV2_0
[==========] Running 1 test from 1 test suite.
[----------] Global test environment set-up.
[----------] 1 test from NxpNfc_DualCpuTest
[ RUN      ] NxpNfc_DualCpuTest.NxpNfc_DualCpu_modeSwitch
Select the option
 1. Switch to EMVCo Mode (Host: SMCU)
 2. Switch to NFC Mode  (Host: Android)
 3. Switch to Secure FW Dnld  (Host: SMCU)
 Please Select : 3
 You selected : 3
 **** Switch to SMCU Host for FW DNLD. Wait for Download to be Completed ****

 **** FW DNLD Completed. Restarting the NFC stack ****

[       OK ] NxpNfc_DualCpuTest.NxpNfc_DualCpu_modeSwitch (32172 ms)
[----------] 1 test from NxpNfc_DualCpuTest (32172 ms total)

[----------] Global test environment tear-down
[==========] 1 test from 1 test suite ran. (32173 ms total)
[  PASSED  ] 1 test.
evk_8mn:/system/lib64 $
```

Figure 33. Secure MCU Mode Switch application output

The FW update has now been completed.

7 Abbreviations and acronyms

Table 2. Abbreviations

Acronym	Description
APDU	Application Protocol Data Unit
AOSP	Android open source project
DH	Device host
HAL	Hardware abstraction layer
FW	Firmware
I^2C	Inter-Integrated Circuit
LPCD	Lower powered card detection
NCI	NFC controller interface
NFC	Near-field communication
MW	Middleware
PLL	Phase-locked loop
P2P	Peer-to-peer
RF	Radio frequency
SDA	Serial data
SMCU	Secure microcontroller
SW	Software

8 References

[1] User guide – UG10068 – PN7220 – Quick start guide (link)
[2] Webpage – PN7220 – EMV L1 Compliant NFC Controller with NCI Interface Supporting EMV and NFC Forum Applications (link)
[3] Kinetis® K82-150 MHz HW Cryptographic Co-Processor and QuadSPI Microcontrollers (MCUs) Based on Arm® Cortex®-M4 Core (link)
[4] Webpage – MCUXpresso Integrated Development Environment (IDE) (link)
[5] Webpage – MCUXpresso SDK Builder (link)
[6] Design resource – NciLib_PUB (link)
[7] User guide – UG10055 – MCUXpresso IDE 24.12 User Guide (link)
[8] Installation Guide – UG10060 – MCUXpresso IDE 24.12 Installation Guide (link)
[9] Webpage – MCU Tech Minutes | Importing an SDK Package into MCUXpresso IDE (link)
[10] Git repository – PN7220 FW GitHub (link)

9 Note about the source code in the document

Example code shown in this document has the following copyright and BSD-3-Clause license:

10 Revision history

Table 3. Revision history

Document ID	Release date	Description
AN14224 v.3.0	7 February 2025	Editorial changes. • Section 1 "Introduction" updated • Section 3 "Architecture" updated • Section 4 "Environment preparation" updated • Section 4.2 "SDK installation" updated • Section 4.3 "Import examples into MCUXpresso" updated • Section 5 "Explanation of applications" updated • Section 6 "Running the examples" updated • Section 6.1 "Secure FW update" added • Section 9 "Note about the source code in the document " updated
AN14224 v.2.0	28 May 2024	• Section 5 "Explanation of applications" updated
AN14224 v.1.0	8 April 2024	• Initial version

Legal information

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.

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 https://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.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

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.

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.

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.

EdgeVerse — is a trademark of NXP B.V.

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

Oracle and Java — are registered trademarks of Oracle and/or its affiliates.

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'.