AN14285

PN7150 to PN7160 Migration Guidelines

Rev. 1.0 — 11 April 2024

Application note

Document information

Information	Content
Keywords	PN7150 NFC Controller, PN7160 NFC Controller, Migration Guidelines, Linux, Android, MCU bare metal
Abstract	This application note describes the guidelines to migrate from the PN7150 NFC Controller to the PN7160 NFC Controller.

1 Introduction

1.1 Purpose

This document provides guidelines for the migration from PN7150 NFC Controller to PN7160 NFC Controller. It aims to describe key differences and new features of PN7160 NFC Controller compared to PN7150 NFC Controller from both a hardware and software perspective.

1.2 Scope

PN7160 NFC Controller is not pin-to-pin compatible with the PN7150 NFC Controller. This migration guide explains how to meet compatibility between PN7150 NFC Controller and PN7160 NFC Controller.

1.3 Audience

This document is intended for customers:

- who have developed their products based on PN7150 NFC Controller and have decided to migrate to the PN7160 NFC Controller.
- who are familiar with PN7150 NFC Controller and want to start their new products based on PN7160 NFC Controller .

2 High-level comparison between PN7150 and PN7160

The PN7150 NFC Controller and PN7160 NFC Controller are NFC controllers designed for a quick integration into a wide range of NFC applications, such as home automation devices or mobile devices. They have been designed for a quick integration for a wide range of systems compliant with NFC standards (NFC Forum, NCI). The software package released by NXP Semiconductors includes drivers for Android and Linux, and supports RTOS and no OS applications.

Full product details, including but not limited to the hardware, reference software and documentation (brochure, data sheet, user manual and application notes), refer to [1] and [2].

PN7150 NFC Controller and PN7160 NFC Controller are similar in terms of features. Table 1 compares the features of both products.

Table 1. Features comparison

Feature	PN7150 NFC Controller	PN7160 NFC Controller
Arm Cortex-M0 core	Yes	Yes
Integrated nonvolatile memory	Yes	Yes
Supported Host Interfaces	I^2C	I^2C and SPI
Host Interface supply voltage	1.8V to 3.3V	1.8V to 3.3V
TX output max current	180mA	250mA
Integrated RF level detector	Yes	Yes
Integrated Polling Loop for automatic device discovery	Yes	Yes
Apple Enhanced Contactless Polling	ECP 1.0*	ECP 1.0 / ECP 2.0**
NFC Standard	NCI 1.0	NCI 2.0
Direct connection to a battery voltage range	(2.3V to 5.5V)	(2.5V to 5.5V)
DPC function	No	Yes
Low power states	No	Yes
Firmware download	No	Yes

** Only for PN7150X version*

*** Only on PN7161*

NFC Forum NFC-IP and Reader modes match for PN7150 NFC Controller and PN7160 NFC Controller, while for card emulation mode there are some differences. See Table 2.

Table 2. Card Emulation Protocol differences

Protocol	PN7150 NFC Controller	PN7160 NFC Controller
T4T - ISO/IEC 14443 A	Yes	Yes
T4T - ISO/IEC 14443 B	Yes	Yes
NFC Forum T3T (FeliCa)	Yes	No

3 Hardware considerations

3.1 Pin-to-pin compatibility

The PN7160 NFC Controller is not pin-to-pin compatible with the PN7150 NFC Controller. When migrating from PN7150 NFC Controller to PN7160 NFC Controller, the next pin configurations that appear on Table 3 must be considered:

Table 3. Pins considerations.

Highlight	PIN	PN7150 NFC Controller	PN7160 NFC Controller	Description
New feature	2	i.c. to ground	DWL_REQ	PN7160 supports NFC controller firmware download using host interface commands. DWL_REQ PIN shall be pulled to VDD(PAD) before reset via VEN pin is done to enter this mode
New feature	3	I2C_ADDR1	I2C_ADDR1/SPI_MOSI	PN7160 supports SPI interface in addition to I2C
New feature	5	I2C_SDA	I2C_SDA/SPI_MISO	PN7160 supports SPI interface in addition to I2C
New feature	7	I2C_SDL	I2C_SDL/SPI_SCK	PN7160 supports SPI interface in addition to I2C
Renamed and range changed	13	VBAT1	VDD(UP)	VBAT1 is the power supply of the LDO for the PN7150. It can be in the range of 2.8V to 5.0V VDD(UP) is the power supply that allows the PN7160 to generate TXLDO for the PN7160. It can be in the range 2.8V to 5.8V
Added PIN (before VDD(TX_IN)) now TVDD_IN and TVDD_IN2	18	TX2	TVDD_IN	TVDD_IN must be connected to VDD(TX) and TVDD_IN2
Relocated	19	VSS(TX)	TX2	Shifted from PIN 18 to 19
Relocated	20	n.c	VSS(TX)	Shifted from PIN 19 to 20
Renamed	22	VDD(TX_IN)	TVDD_IN2	This pin must be connected to VDD(TX) in PN7150. In PN7160 this PIN must be connected to VDD(TX) and TVDD_IN.
New feature	23	i.c	ANT_1	PN7160 implements a very-low power RF detector. To activate this function the antenna has to be connected to pins ANT1 and ANT2
New feature	24	i.c	ANT_2	PN7160 implements a very-low power RF detector. To activate this function the antenna has to be connected to pins ANT1 and ANT2
New feature	25	i.c	VDD_HF	VDD(HF) monitors the rectifier output for the very-low power RF detector feature
Relocated	26	VBAT	VDD(A)	VDD(A) is the analog supply voltage. Changed from pin 28 to pin 26. Must be connected to VDD(D)
Relocated	27	VSS	VDD	VDD changed from pin 29 to pin 26. Must be connected to AVDD and DVDD

Table 3. Pins considerations*...continued*

Highlight	PIN	PN7150 NFC Controller	PN7160 NFC Controller	Description
Relocated	28	VDD(A)	VBAT	VBAT changes from pin 26 to pin 28. It is the battery supply voltage and must be connected to VBAT2 in the PN7160
Relocated	29	VDD	XTAL2	Oscillator output. Changed from pin 37 to pin 29
Relocated	30	VDD(D)	NFC_CLK_XTAL_1	NFC_CLK_XTAL1 is the PLL input. Changed from pin 36 to pin 30
Relocated	31	n.c.	VDD(D)	VDD(D) is the digital supply voltage. Must be connected to VDD and VDD(A). It changes from pin 30 to pin 31
Relocated	36	NFC_CLK_XTAL_1	n.c.	This was the PLL input. Now, it must be left unconnected
New feature	37	NFC_CLK_XTAL_2	DCDC_EN	External DC-DC enable request on VDD(PAD)
New feature	39	i.c.	WUP_REQ	It allows the PN7160 to wake-up from standby through a host using this pin

** i.c: internally connected; leave open.*

** n.c: not connected.*

3.2 Packaging information

PN7160 NFC Controller is available in two packaging configurations: VFBGA64 and HVQFN40, while PN7150 NFC Controller is only available in HVQFN40.

PN7150 NFC Controller and PN7160 NFC Controller are not pin-to-pin compatible in HVQFN40 package version. To see the differences check Section 3.1.

For more information about package specifications, please refer to the datasheets ([3], [4]) and hardware design guides ([5], [6]) of PN7150 NFC Controller and PN7160 NFC Controller, respectively.

3.3 NFC antenna matching

The NFC matching circuit architecture of the PN7150 NFC Controller and the PN7160 NFC Controller are the same; however, two things have to be considered:

1. As the PN7160 NFC Controller allows higher TX output current than the PN7150, the matching impedance value to be targeted can be lower for the PN7160 compared to the PN7150.
2. Two capacitors can be optionally connected to the ANT pins of the PN7160 NFC Controller for low power RF field detector. The PN7150 does not implement this feature.

Figure 1 below shows the way to connect the differential antenna for the PN7160 NFC Controller.

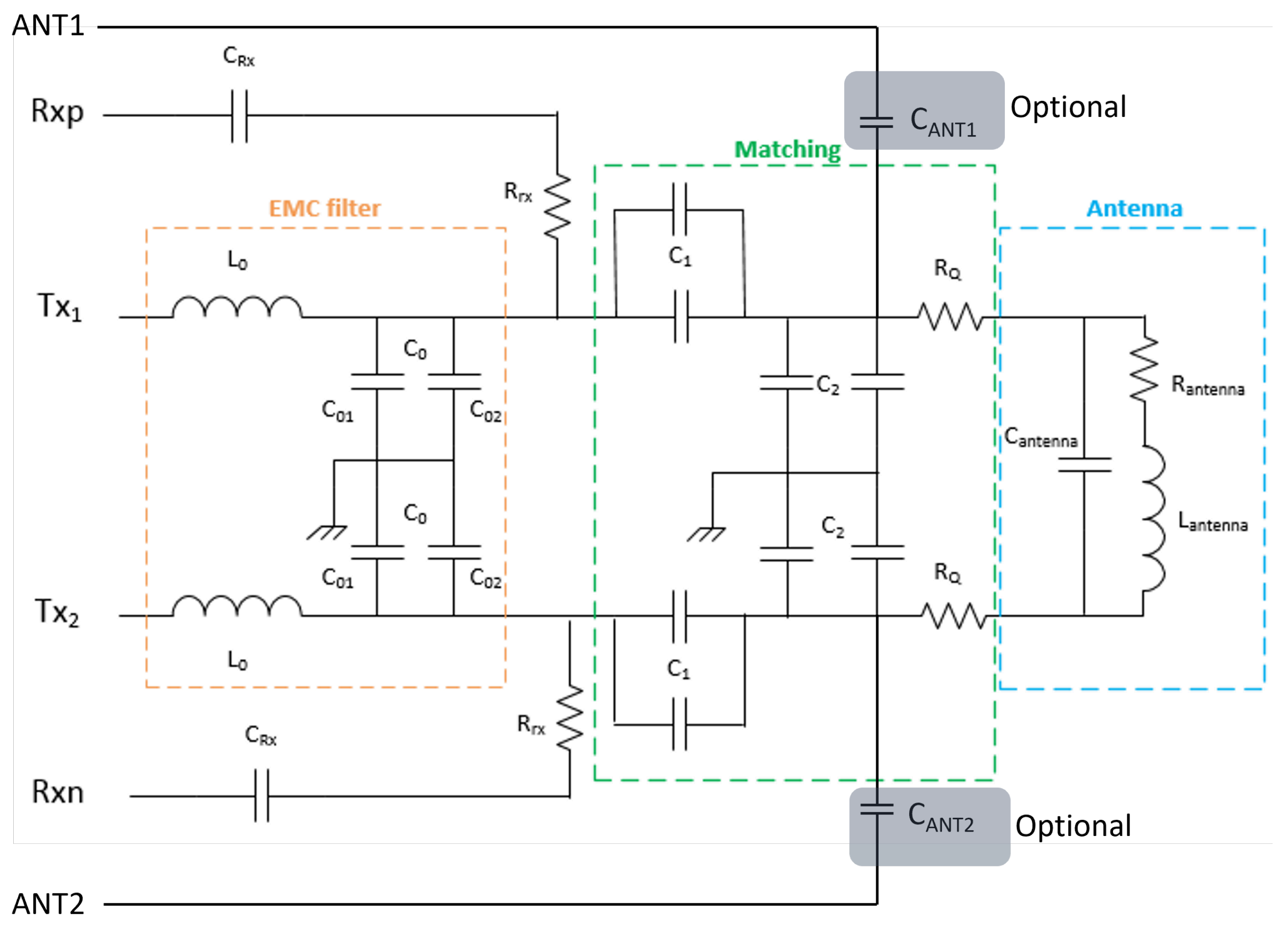


Figure 1. Antenna matching circuit

The schematic above shows the architecture for the PN7160 NFC Controller with the ANT1 and ANT2 pins.

Find all the information about the new antenna matching circuit for the PN7160 NFC Controller on the PN7160 antenna design and matching guide

4 Software considerations

This section details how to migrate software projects that integrate support for the PN7150 NFC Controller to support the PN7160 NFC Controller. MCU BareMetal, Linux, and Android software projects are considered in this document.

Figure 2 shows the connection diagram for both PN7150 NFC Controller and PN7160 NFC Controller. As can be observed, the application processor relies on the host interface to transmit NCI commands to the NFC controller. The NFC controller firmware running on the device is responsible for communicating with the contactless products through the NFC antenna. This means that the hardware differences detailed in Section 3.1, such as the different PINs for TX2, NFC_CLK_XTAL1 and NFC_CLK_XTAL2 symbols, are transparent to the software projects. The NFC controller firmware itself will be responsible to manage the NFC antenna on its corresponding PINs.

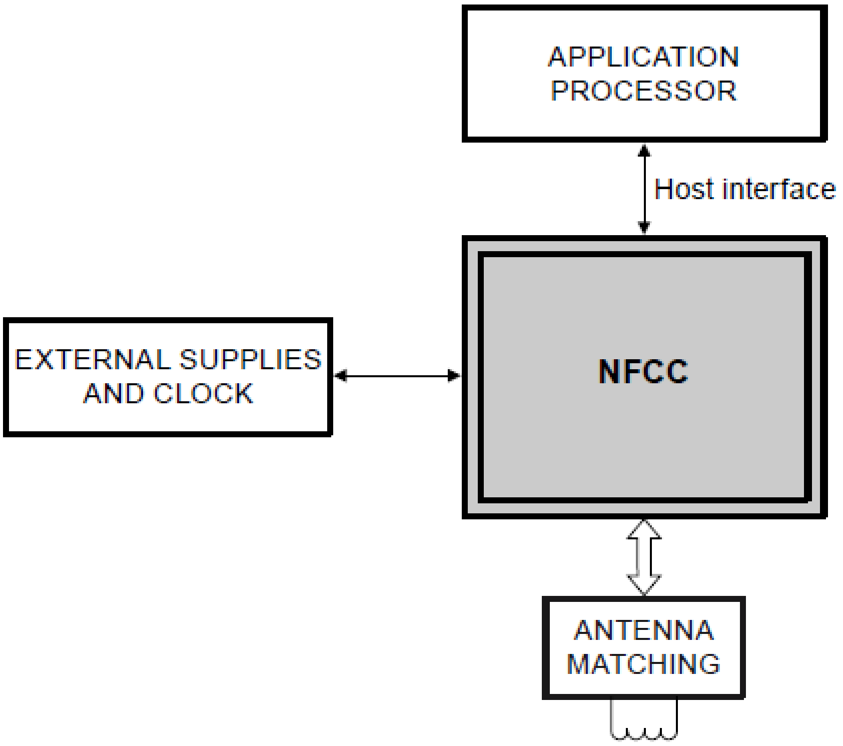


Figure 2. PN71x0 connection

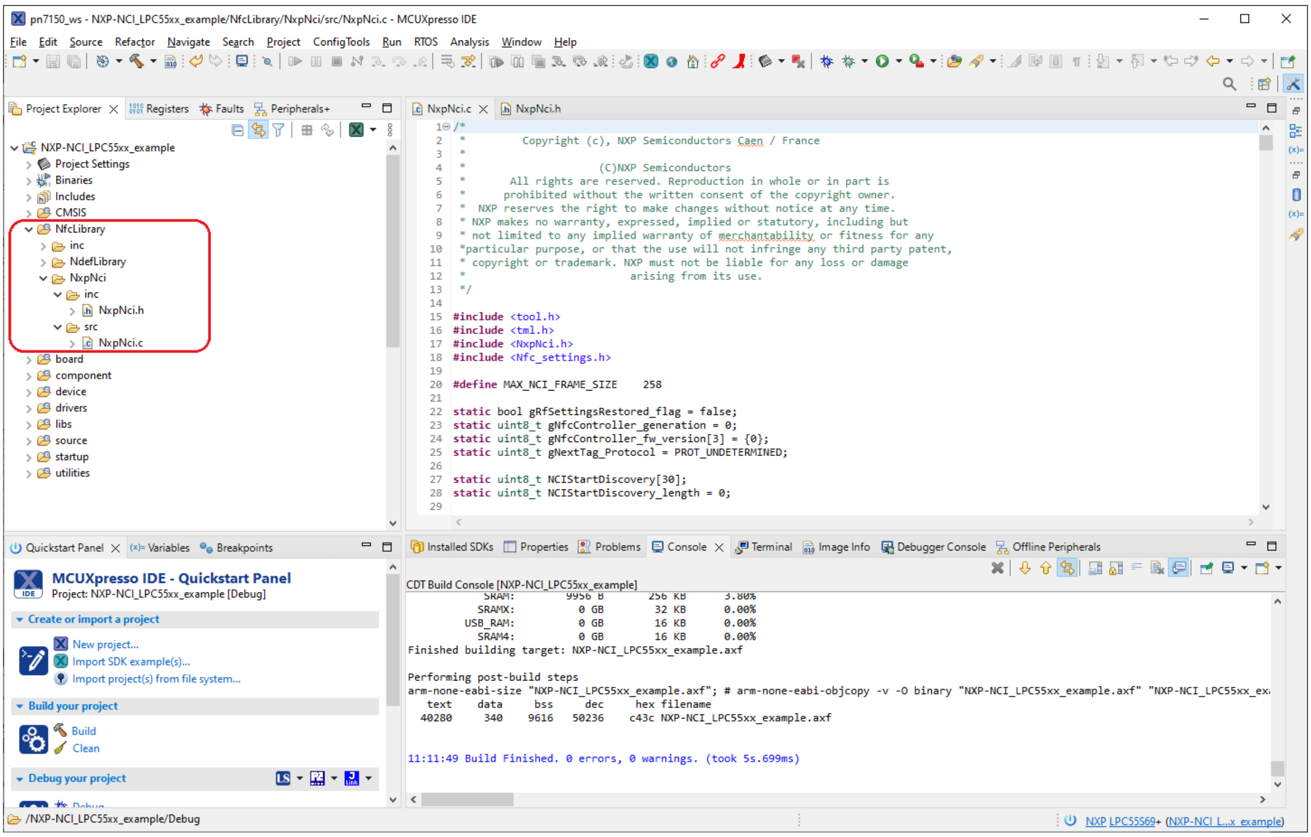
As explained in Section 2, PN7150 NFC Controller supports NCI 1.0 while PN7160 NFC Controller supports NCI 2.0, which is not fully backward compatible.

4.1 MCU BareMetal software considerations

This section describes the steps required to migrate NXP's reference MCU BareMetal software SW4325 - PN7150 NXP-NCI MCUXpresso example Project available for PN7150 NFC Controller to support PN7160 NFC Controller. The reference SW6705 - PN7160 NXP-NCI MCUXpresso Example Project provides reference source code for PN7160 NFC Controller.

Reference MCU BareMetal software projects used to validate the NFC controller migration are used NXP's LPC55S6x MCU family as the host device, but the migration steps detailed here are agnostic of the MCU used by the NFC Reader manufacturer.

The SW4325 - PN7150 NXP-NCI MCUXpresso example Project taken as the reference MCU BareMetal software project for the PN7150 NFC Controller integrates NXP's NfcLibrary folder, which provides NCI 1.0 implementation (see Figure 3).



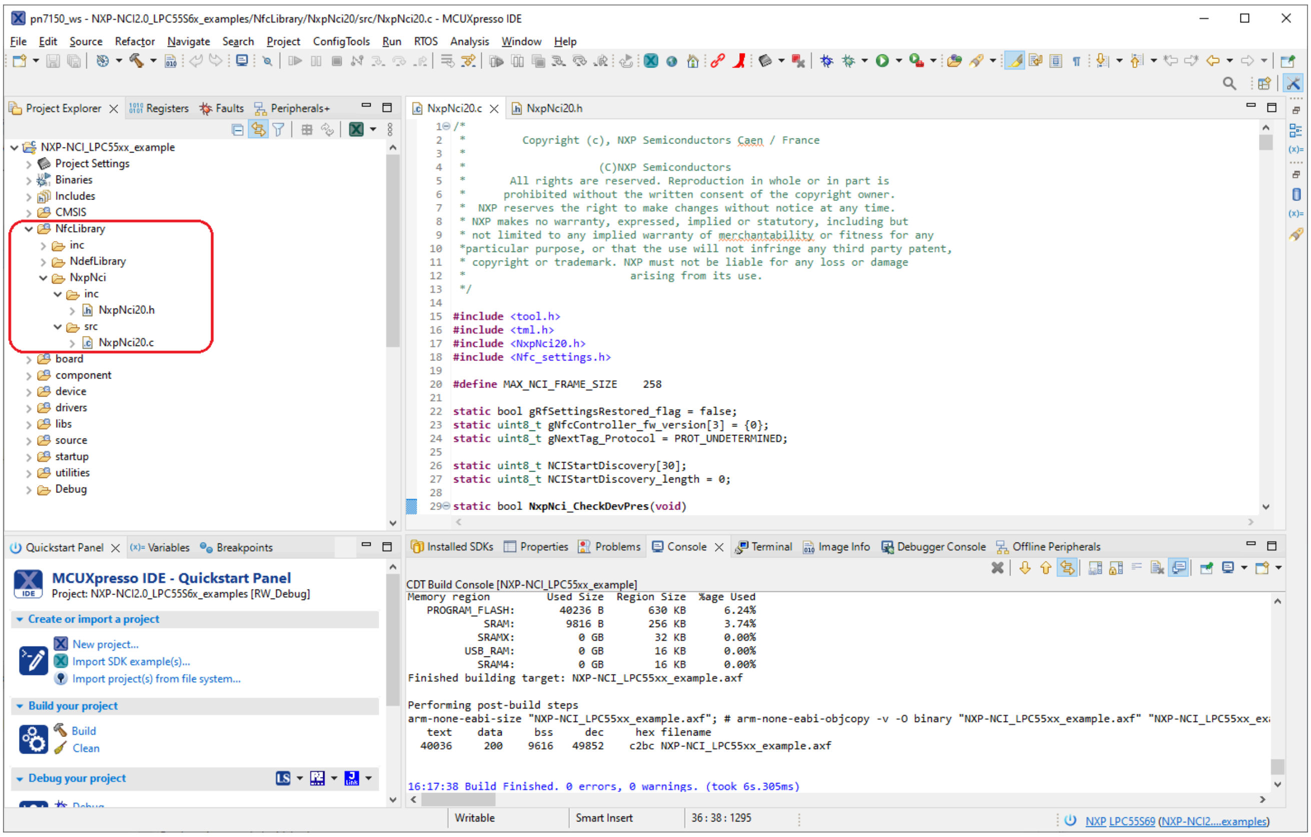
```
/*
 *         Copyright (c), NXP Semiconductors Caen / France
 *
 *                     (C)NXP Semiconductors
 *       All rights are reserved. Reproduction in whole or in part is
 *      prohibited without the written consent of the copyright owner.
 *  NXP reserves the right to make changes without notice at any time.
 * NXP makes no warranty, expressed, implied or statutory, including but
 * not limited to any implied warranty of merchantability or fitness for any
 *particular purpose, or that the use will not infringe any third party patent,
 * copyright or trademark. NXP must not be liable for any loss or damage
 *                          arising from its use.
 */

#include <tool.h>
#include <tml.h>
#include <NxpNci.h>
#include <Nfc_settings.h>

#define MAX_NCI_FRAME_SIZE    258

static bool gRfSettingsRestored_flag = false;
static uint8_t gNfcController_generation = 0;
static uint8_t gNfcController_fw_version[3] = {0};
static uint8_t gNextTag_Protocol = PROT_UNDETERMINED;

static uint8_t NCIStartDiscovery[30];
static uint8_t NCIStartDiscovery_length = 0;

```

CDT Build Console [NXP-NCI_LPC55xx_example]

```
          SRAM:        9956 B       256 KB      3.80%
         SRAMX:           0 GB        32 KB      0.00%
       USB_RAM:           0 GB        16 KB      0.00%
         SRAM4:           0 GB        16 KB      0.00%
Finished building target: NXP-NCI_LPC55xx_example.axf

Performing post-build steps
arm-none-eabi-size "NXP-NCI_LPC55xx_example.axf"; # arm-none-eabi-objcopy -v -O binary "NXP-NCI_LPC55xx_example.axf" "NXP-NCI_LPC55xx_exa
   text    data     bss     dec     hex filename
  40280     340    9616   50236    c43c NXP-NCI_LPC55xx_example.axf


11:11:49 Build Finished. 0 errors, 0 warnings. (took 5s.699ms)
```

Figure 3. PN7150 SW - NFC NCI Library 1.0

As explained previously, the PN7160 NFC Controller supports NCI 2.0, which is not fully backward compatible with NCI 1.0. Therefore, to support the PN7160 NFC Controller, the content of the NfcLibrary folder must be replaced with the content of the NfcLibrary folder available in SW6705 - PN7160 NXP-NCI MCUXpresso Example Project, which provides NCI 2.0 implementation (see Figure 4).

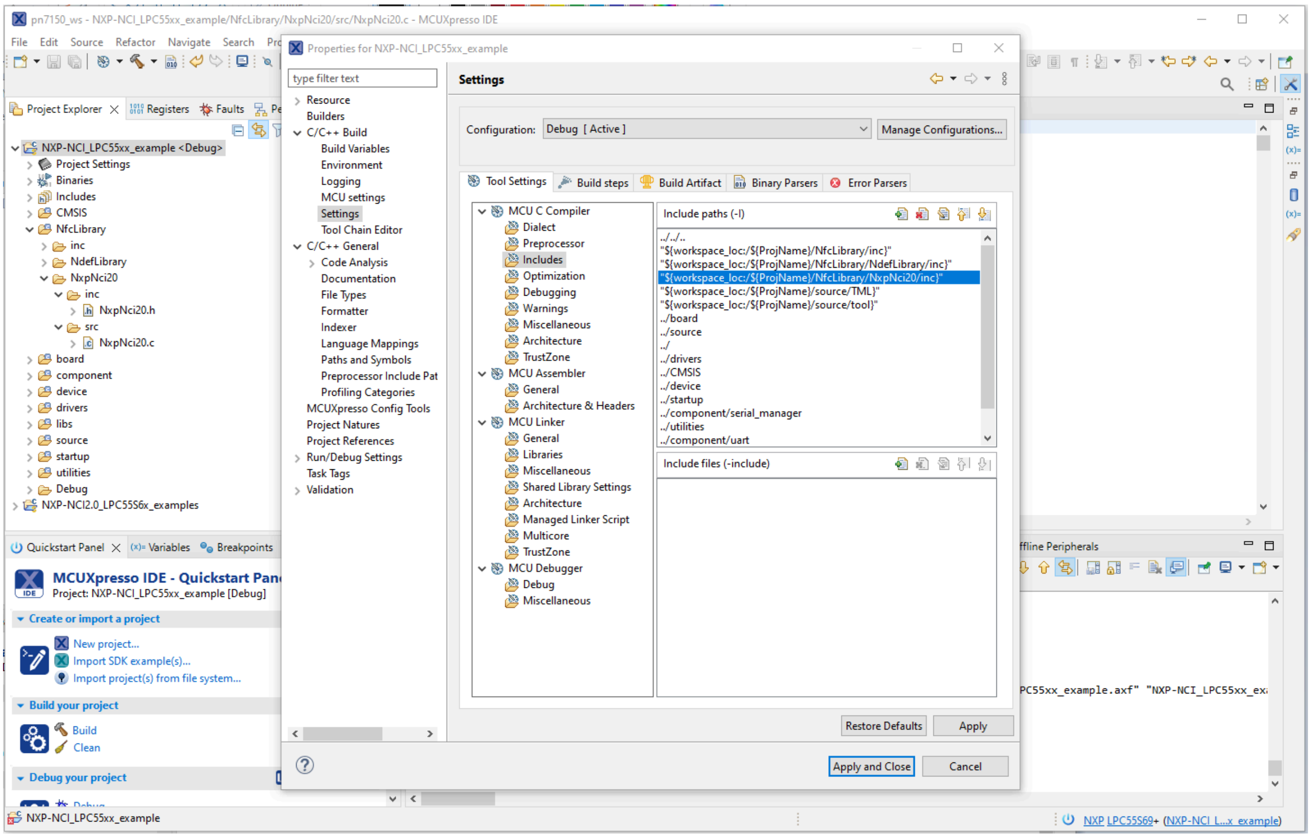

pn7150_ws - NXP-NCI2.0_LPC55S6x_examples/NfcLibrary/NxpNci20/src/NxpNci20.c - MCUXpresso IDE
File Edit Source Refactor Navigate Search Project ConfigTools Run RTOS Analysis Window Help
Project Explorer Registers Faults Peripherals+
NXP-NCI_LPC55xx_example
Project Settings
Binaries
Includes
CMSIS
NfcLibrary
inc
NdefLibrary
NxpNci
inc
NxpNci20.h
src
NxpNci20.c
board
component
device
drivers
libs
source
startup
utilities
Debug
NxpNci20.c
NxpNci20.h
Quickstart Panel Variables Breakpoints
MCUXpresso IDE - Quickstart Panel
Project: NXP-NCI2.0_LPC55S6x_examples [RW_Debug]
Create or import a project
New project...
Import SDK example(s)...
Import project(s) from file system...
Build your project
Build
Clean
Debug your project
Installed SDKs Properties Problems Console Terminal Image Info Debugger Console Offline Peripherals
CDT Build Console [NXP-NCI_LPC55xx_example]
Writable
Smart Insert
36 : 38 : 1295
NXP LPC55S69 (NXP-NCI2....examples)

```
/*
 *          Copyright (c), NXP Semiconductors Caen / France
 *
 *                      (C)NXP Semiconductors
 *        All rights are reserved. Reproduction in whole or in part is
 *       prohibited without the written consent of the copyright owner.
 *   NXP reserves the right to make changes without notice at any time.
 * NXP makes no warranty, expressed, implied or statutory, including but
 * not limited to any implied warranty of merchantability or fitness for any
 *particular purpose, or that the use will not infringe any third party patent,
 * copyright or trademark. NXP must not be liable for any loss or damage
 *                          arising from its use.
 */

#include <tool.h>
#include <tml.h>
#include <NxpNci20.h>
#include <Nfc_settings.h>

#define MAX_NCI_FRAME_SIZE    258

static bool gRfSettingsRestored_flag = false;
static uint8_t gNfcController_fw_version[3] = {0};
static uint8_t gNextTag_Protocol = PROT_UNDETERMINED;

static uint8_t NCIStartDiscovery[30];
static uint8_t NCIStartDiscovery_length = 0;

static bool NxpNci_CheckDevPres(void)
```

```
Memory region         Used Size  Region Size  %age Used
   PROGRAM_FLASH:       40236 B       630 KB      6.24%
            SRAM:        9816 B       256 KB      3.74%
           SRAMX:          0 GB        32 KB      0.00%
         USB_RAM:          0 GB        16 KB      0.00%
           SRAM4:          0 GB        16 KB      0.00%
Finished building target: NXP-NCI_LPC55xx_example.axf

Performing post-build steps
arm-none-eabi-size "NXP-NCI_LPC55xx_example.axf"; # arm-none-eabi-objcopy -v -O binary "NXP-NCI_LPC55xx_example.axf" "NXP-NCI_LPC55xx_ex
   text    data     bss     dec     hex filename
  40036     200    9616   49852    c2bc NXP-NCI_LPC55xx_example.axf


16:17:38 Build Finished. 0 errors, 0 warnings. (took 6s.305ms)
```

Figure 4. PN7160 SW - NFC NCI Library 2.0 update

It is required to update MCUXpresso software project properties to set the path to the new NxpNci20 folder in the NfcLibrary folder (see Figure 5).



Configuration: Debug [Active]

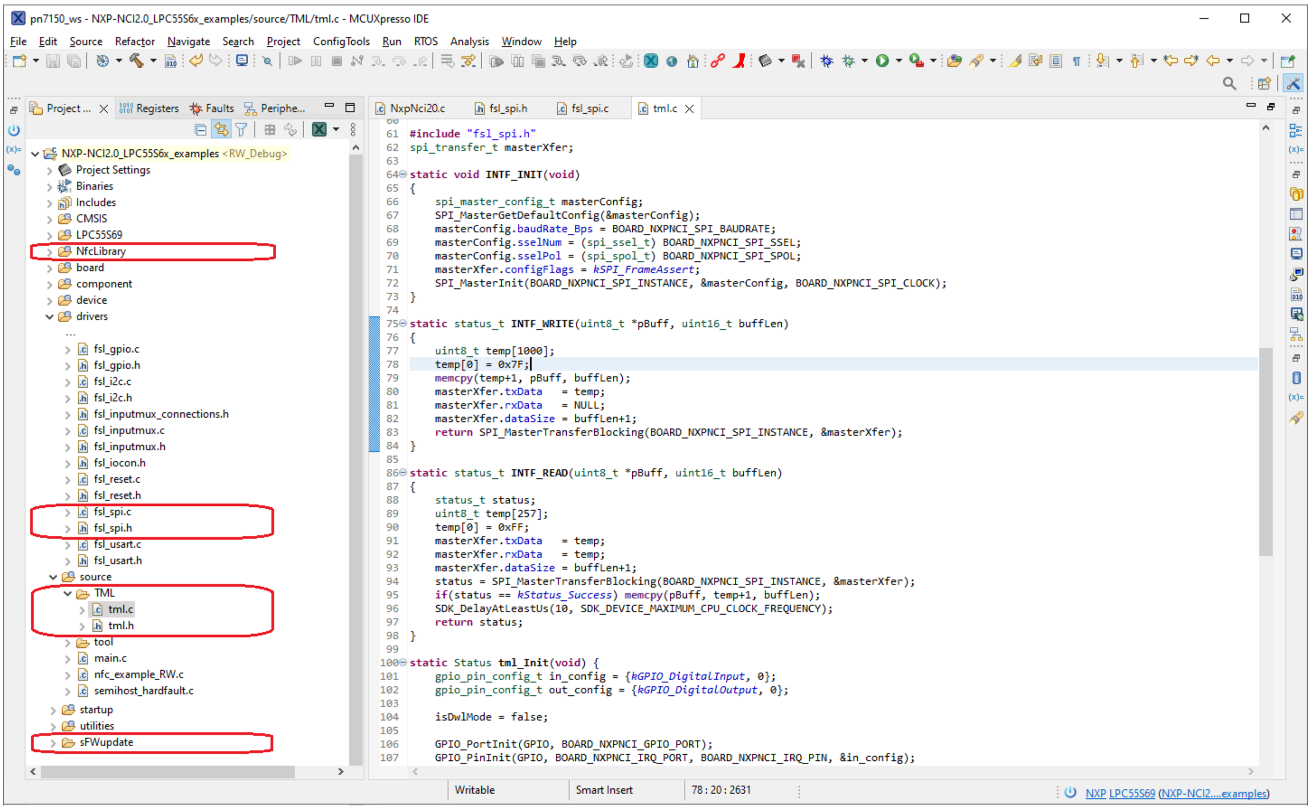
Include paths (-I)

```
../../..
"${workspace_loc:/${ProjName}/NfcLibrary/inc}"
"${workspace_loc:/${ProjName}/NfcLibrary/NdefLibrary/inc}"
"${workspace_loc:/${ProjName}/NfcLibrary/NxpNci20/inc}"
"${workspace_loc:/${ProjName}/source/TML}"
"${workspace_loc:/${ProjName}/source/tool}"
../board
../source
../
../drivers
../CMSIS
../device
../startup
../component/serial_manager
../utilities
../component/uart
```

Figure 5. PN7160 SW - MCUXpresso software project properties

If your PN7150 MCU software project supports NFC Forum T3T (FeliCa) Card Emulation, this needs to be removed because, as explained in Section 2, PN7160 NFC Controller does not support this mode.

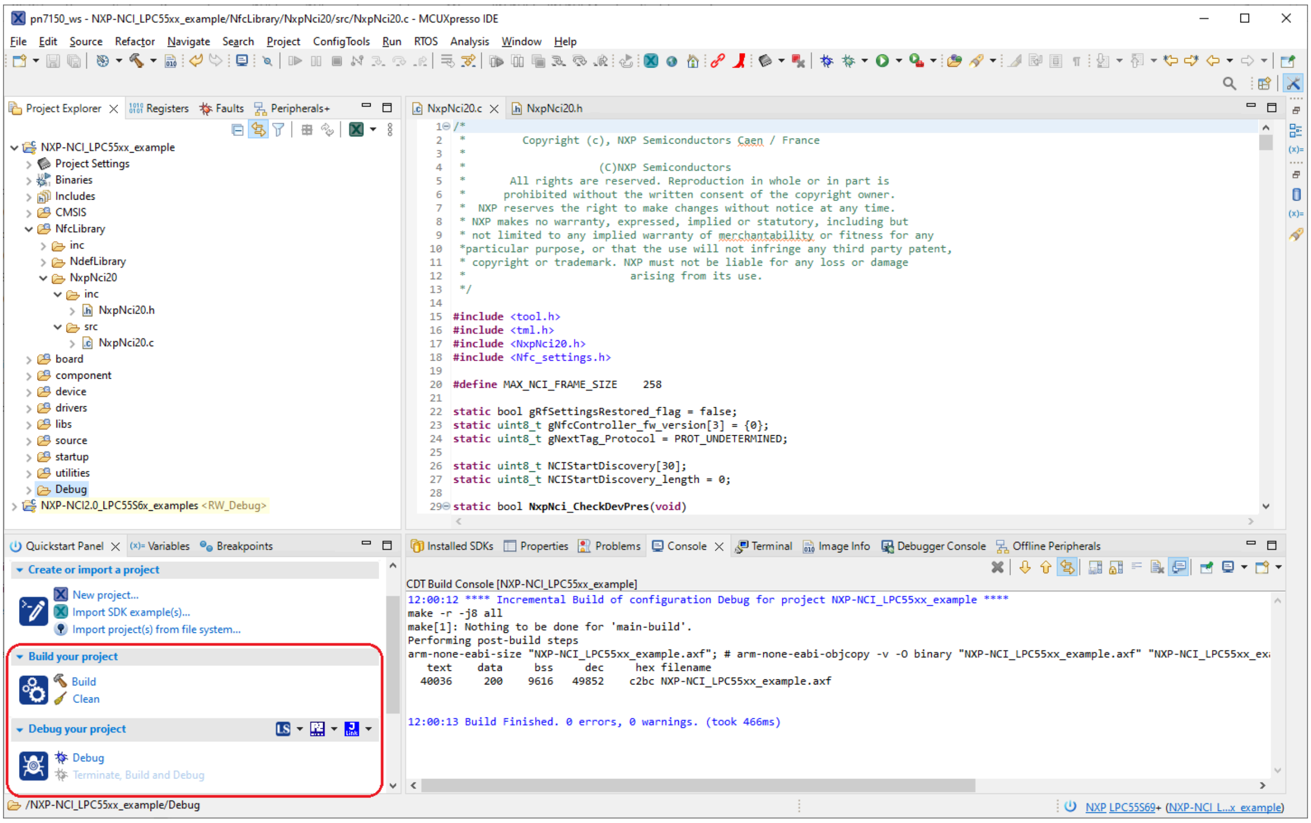
In addition to the steps described previously, which ensure that a legacy PN7150 NFC Controller MCU BareMetal software project can be executed on a PN7160 NFC Controller, it is highly recommended to integrate the new features supported by PN7160 NFC Controller that are available in SW6705 - PN7160 NXP-NCI MCUXpresso Example Project. These features include, among others, the SPI host interface support, which can be found in drivers folder and TML implementation files; the firmware download support, which can be found in a dedicated folder; and the integration of the new for low-power mode supported at NCI level (see Figure 6). PN7160 card emulation document provides full details and software examples on how to properly set Card Emulation (CE) for specific CE scenario in an MCUXpresso project.



```
#include "fsl_spi.h"
spi_transfer_t masterXfer;

static void INTF_INIT(void)
{
    spi_master_config_t masterConfig;
    SPI_MasterGetDefaultConfig(&masterConfig);
    masterConfig.baudRate_Bps = BOARD_NXPNCI_SPI_BAUDRATE;
    masterConfig.sselNum = (spi_ssel_t) BOARD_NXPNCI_SPI_SSEL;
    masterConfig.sselPol = (spi_spol_t) BOARD_NXPNCI_SPI_SPOL;
    masterXfer.configFlags = kSPI_FrameAssert;
    SPI_MasterInit(BOARD_NXPNCI_SPI_INSTANCE, &masterConfig, BOARD_NXPNCI_SPI_CLOCK);
}

static status_t INTF_WRITE(uint8_t *pBuff, uint16_t buffLen)
{
    uint8_t temp[1000];
    temp[0] = 0x7F;
    memcpy(temp+1, pBuff, buffLen);
    masterXfer.txData   = temp;
    masterXfer.rxData   = NULL;
    masterXfer.dataSize = buffLen+1;
    return SPI_MasterTransferBlocking(BOARD_NXPNCI_SPI_INSTANCE, &masterXfer);
}

static status_t INTF_READ(uint8_t *pBuff, uint16_t buffLen)
{
    status_t status;
    uint8_t temp[257];
    temp[0] = 0xFF;
    masterXfer.txData   = temp;
    masterXfer.rxData   = temp;
    masterXfer.dataSize = buffLen+1;
    status = SPI_MasterTransferBlocking(BOARD_NXPNCI_SPI_INSTANCE, &masterXfer);
    if(status == kStatus_Success) memcpy(pBuff, temp+1, buffLen);
    SDK_DelayAtLeastUs(10, SDK_DEVICE_MAXIMUM_CPU_CLOCK_FREQUENCY);
    return status;
}

static Status tml_Init(void) {
    gpio_pin_config_t in_config = {kGPIO_DigitalInput, 0};
    gpio_pin_config_t out_config = {kGPIO_DigitalOutput, 0};

    isDwlMode = false;

    GPIO_PortInit(GPIO, BOARD_NXPNCI_GPIO_PORT);
    GPIO_PinInit(GPIO, BOARD_NXPNCI_IRQ_PORT, BOARD_NXPNCI_IRQ_PIN, &in_config);
```

Figure 6. PN7160 SW - New features support

Finally, and once the MCUXpresso software project is successfully built, the image binary can be flashed into the PN7160 NFC Controller and debugged by clicking the Debug button of the MCUXpresso IDE (see Figure 7).



```
/*
 *         Copyright (c), NXP Semiconductors Caen / France
 *
 *                      (C)NXP Semiconductors
 *       All rights are reserved. Reproduction in whole or in part is
 *      prohibited without the written consent of the copyright owner.
 *  NXP reserves the right to make changes without notice at any time.
 * NXP makes no warranty, expressed, implied or statutory, including but
 * not limited to any implied warranty of merchantability or fitness for any
 *particular purpose, or that the use will not infringe any third party patent,
 * copyright or trademark. NXP must not be liable for any loss or damage
 *                          arising from its use.
 */

#include <tool.h>
#include <tml.h>
#include <NxpNci20.h>
#include <Nfc_settings.h>

#define MAX_NCI_FRAME_SIZE    258

static bool gRfSettingsRestored_flag = false;
static uint8_t gNfcController_fw_version[3] = {0};
static uint8_t gNextTag_Protocol = PROT_UNDETERMINED;

static uint8_t NCIStartDiscovery[30];
static uint8_t NCIStartDiscovery_length = 0;

static bool NxpNci_CheckDevPres(void)
```

CDT Build Console [NXP-NCI_LPC55xx_example]

```
12:00:12 **** Incremental Build of configuration Debug for project NXP-NCI_LPC55xx_example ****
make -r -j8 all
make[1]: Nothing to be done for 'main-build'.
Performing post-build steps
arm-none-eabi-size "NXP-NCI_LPC55xx_example.axf"; # arm-none-eabi-objcopy -v -O binary "NXP-NCI_LPC55xx_example.axf" "NXP-NCI_LPC55xx_exa
   text    data     bss     dec     hex filename
  40036     200    9616   49852    c2bc NXP-NCI_LPC55xx_example.axf


12:00:13 Build Finished. 0 errors, 0 warnings. (took 466ms)
```

Figure 7. PN7160 SW - MCUXpresso software project debug

4.2 Linux software considerations

This section details how to migrate a Linux platform project that integrates support for the PN7150 NFC Controller to the PN7160 NFC Controller. [7] and [8] provide full guidelines for the integration of PN7150 and PN7160 NXP NCI-based NFC controllers to a Linux platform and, therefore, these details will not be covered in this document. It is assumed that the reader has already cloned the Linux project, added NFC support for PN7150 NFC Controller and built image successfully.

The process described in this document is not distribution dependent and should progress as what follows. The reference library and firmware that will be used in this document can be found in these two GitHub repositories for PN7150 NFC Controller and PN7160 NFC Controller.

Figure 8 (to be found in [8]) shows the library in the Linux NFC stack that will be updated in the following section of this document to migrate the Linux platform to support the PN7160 NFC Controller.

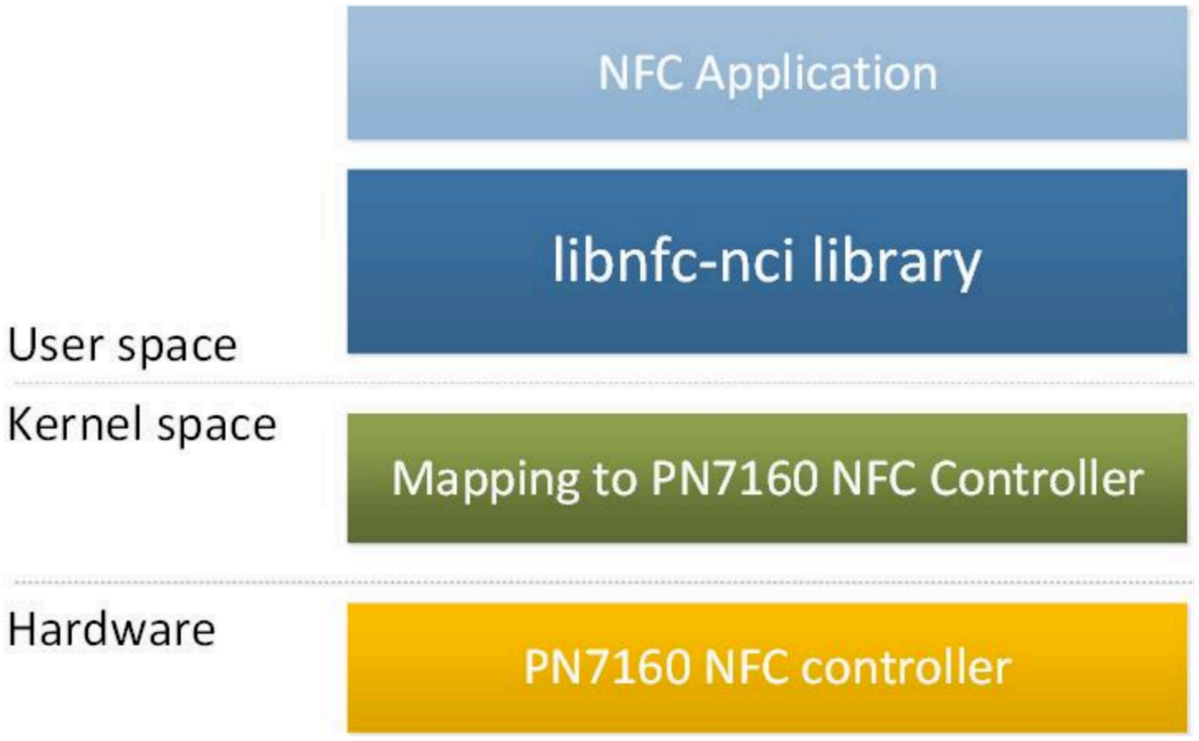


Figure 8. PN7160 Linux libnfc-nci stack software

4.2.1 Kernel driver

4.2.1.1 Kernel driver nxpnfc

The Linux NFC stack can use NXPs kernel driver to communicate with NCI-based NFC controllers made by NXP.

While the legacy PN7150 NFC Controller I^{2}C kernel driver should function with the PN7160 NFC Controller, it is recommended to update the kernel driver to integrate the newer kernel driver version to allow support for SPI interface. The PN7160 NFC open source kernel driver supporting both I2C and SPI can be found here. [8] explains how to download, integrate, and build the NFC kernel driver into the Linux platform.

4.2.1.2 Alternative to nxpnfc kernel driver

An alternative is possible if the GPIO and I2C & SPI resources can be accessed through the user space. [8] explains how to enable the rights for the application.

4.2.2 NFC library

As already introduced in Section 4.2, the original NCI 1.0-based Linux NFC stack must be modified with the new source code to fully support NCI 2.0-based NFC controllers. The PN7160 NFC Controller supports NFC controller firmware update, which is not supported by the PN7150 NFC Controller, and, therefore, will also need to be added.

The folders to be downloaded are found in this repository.

The folders to be replaced are:

- *conf*
- *src*

The new folder to be added is:

- *firmware*

There are minor API changes that must be considered by the NFC Reader manufacturer when replacing the *src* folder that includes the NCI 2.0-based Linux NFC stack that targets the PN7160 NFC Controller.

The API changes are:

- *nfcManager_doInitialize()* becomes *doInitialize()*
- *nfcManager_enableDiscovery(...)* becomes *doEnableDiscovery(...)*
- *nfcManager_disableDiscovery()* becomes *disableDiscovery()*
- *nfcManager_getNumTags()* becomes *getNumTags()*
- *nfcManager_selectNextTag()* becomes *selectNextTag()*
- *nfcManager_registerTagCallback(...)* becomes *registerTagCallback(...)*
- *nfcManager_deregisterTagCallback()* becomes *deregisterTagCallback()*

4.2.3 Build and install

Once all the steps detailed in this section have been completed, the user shall proceed to build the updated library with full support for the PN7160 NFC Controller and install it in the targeted devices. The user needs to adapt paths in the following files to ensure the success of this step. The files to be modified are as follows:

- *Makefile.am*
- *configure.ac*

[8] explains the build and install process for the library.

4.2.4 Verify the NFC functionality

The last step to finalize the migration is to verify the correct functioning of the NFC technology in the Linux device now integrating the PN7160 NFC Controller.

The PN7160 Linux libnfc-nci stack provides an Example application, which can be downloaded from repository, demonstrates the use of the library to run NFC features such as polling or writing NFC tags. [8] explains how to use it.

4.3 Android software considerations

This section details how to migrate an Android platform project that integrates support for the PN7150 NFC Controller to the PN7160 NFC Controller. [9] and [10] provide full guidelines for the integration of PN7150 and PN7160 NXP NCI-based NFC controllers to an Android platform and, therefore, these details will not be covered in this document. It is assumed that the reader has already cloned the Android AOSP project, added NFC support for PN7150 NFC Controller and built the image successfully.

This document focuses on the Android AOSP 11.0 version as this is the latest Android AOSP version supported for PN7150 NFC Controller. Porting to another Android version may require minor adaptation. The reference firmware, configuration files and patches that are used in this document can be found in these two GitHub repositories for PN7150 NFC Controller and PN7160 NFC Controller. When the migration to PN7160 NFC Controller is completed, the NFC Reader manufacturer might consider moving to a higher Android AOSP version.

Figure 9 (to be found in [10]) shows the components in the Android NFC stack that will be updated in the following sections of this document to migrate the Android platform to support the PN7160 NFC Controller.

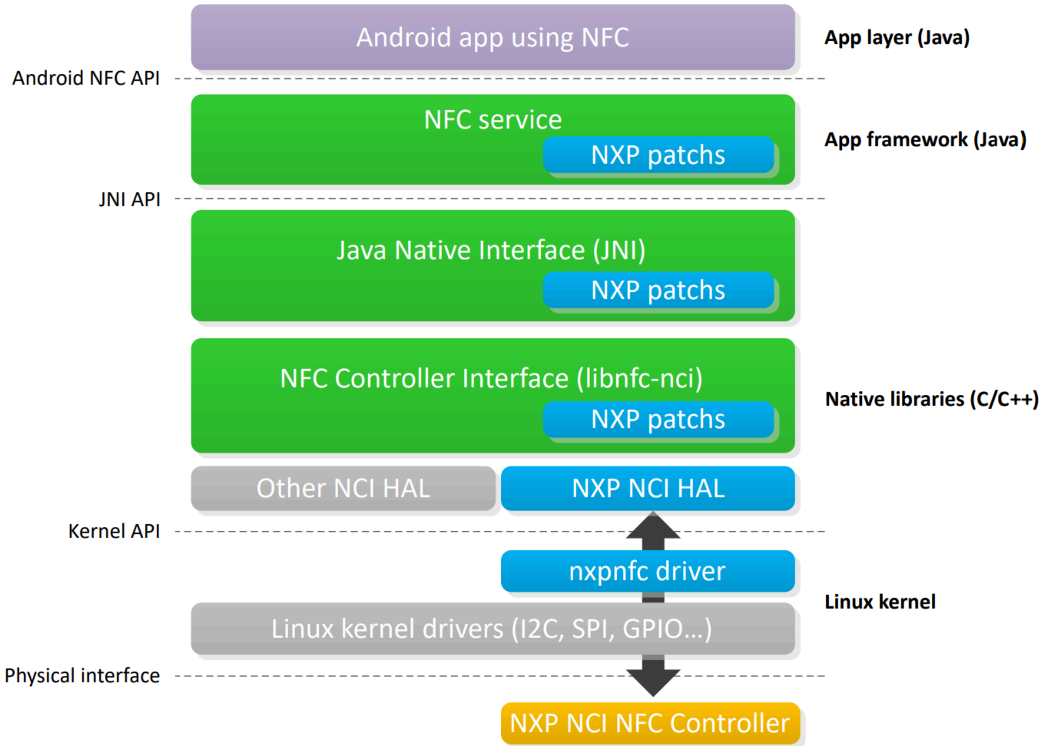


Figure 9. Android NFC stack overview

4.3.1 Kernel driver

The Android NFC stack uses NFC kernel driver to communicate with the NCI-based NFC controller.

Legacy PN7150 NFC Controller I2C kernel driver should be valid to communicate with the PN7160 NFC Controller; however it is highly recommended to update the full NFC kernel driver to integrate the latest available I2C kernel driver version and to support the SPI interface. The PN7160 NFC open source kernel driver supporting both I2C and SPI can be found here. [10] explains how to download, integrate, and build the NFC kernel driver into the Android AOSP platform.

4.3.2 NXP-NCI library

As already introduced in Section 4.3, the original Android AOSP NFC stack must be modified with extensions to fully support NCI-based NFC controllers. NXP provides the necessary patches that must be applied to the corresponding Android AOSP versions to perform this update.

The patches for the PN7160 NFC Controller can be found in this repository.

In this document it is considered that the NFC Reader manufacturer has already applied to the original Android AOSP NFC stack the corresponding patches to support the PN7150 NFC Controller. Therefore, the patches to be used to support the PN7160 NFC Controller will not work out of the box. It is recommended to replace the previously modified folders with the original Android AOSP NFC stack and then apply the new patches for the PN7160 NFC Controller. The list of folders that were modified and that must be restored are:

- *hardware/nxp/nfc*
- *packages/apps/Nfc*
- *frameworks/base*
- *frameworks/native*
- *vendor/nxp*
- *system/nfc*

If the full restoration of the original Android AOSP NFC stack is not possible, the NFC Reader manufacturer is responsible to process the referenced patches one by one and to perform the required changes, which include not only updating those files that were already modified, but also creating those files that are specific to the PN7160 NFC Controller and that did not exist before.

4.3.3 Configuration files

Both *libnfc-nci.conf* and *libnfc-nxp.conf* configuration files in the *vendor/nxp/nfc/hw/* folder must be updated to support the PN7160 NFC Controller.

The reference configuration file for the PN7160 NFC Controller can be found in this repository. It is the responsibility of the NFC Reader manufacturer to fine-tune the configuration files with the integration specifics such as RF settings for the NFC antenna, the clock configuration, etc. Full details regarding the configuration file definition for the PN7160 NFC Controller can be found in Section 6 of [10].

4.3.4 Firmware library

The PN7160 NFC Controller supports NFC controller firmware update, which is not supported by the PN7150 NFC Controller.

The latest PN7160 NFC Controller firmware library, available for both 32 bits and 64-bits platforms, can be downloaded from this repository. The binaries shall be copied to */vendor/nxp/pn7160/firmware* folder for the Android NFC stack to proceed to update the PN7160 NFC Controller firmware at boot-up.

4.3.5 Build and install

Once all the steps detailed in this section have been completed, the user shall proceed to build the updated Android image with full support for the PN7160 NFC Controller and install it in the targeted devices.

4.3.6 Verify the NFC functionality

The last step to finalize the migration is to verify the correct functioning of the NFC technology in the Android device now integrating the PN7160 NFC Controller.

NFC TagInfo by NXP and ***NFC TagWriter by NXP*** are two applications available for free in Google Play that can be used to verify. Read and Write operations with NFC tags.

5 References

[1] Webpage - PN7150: High-Performance NFC Controller with Integrated Firmware for Smart Devices (link)

[2] Webpage - PN7160: NFC Plug and Play Controller with Integrated Firmware and NCI Interface (link)

[3] Data sheet - PN7150: High performance NFC controller with integrated firmware, supporting all NFC Forum modes (link)

[4] Data sheet - PN7160_PN7161: Near Field Communication (NFC) controller (link)

[5] Application note - AN11756: PN7150 Hardware Design Guide (link)

[6] Application note - AN12988: PN7160 hardware design guide (link)

[7] Application note - AN11697: PN71xx Linux Software Stack Integration Guidelines (link)

[8] Application note - AN13287: PN7160 Linux porting guide (link)

[9] Application note - AN11690: NXP NCI Android Porting Guidelines (link)

[10] Application note - AN13189: PN7160 Android porting guide (link)

6 Revision history

Table 4. Revision history

Document ID	Release date	Description
AN14285 v.1.0	11 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.

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.

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: https://www.nxp.com

Date of release: 11 April 2024
Document identifier: AN14285