AN13287

PN7160 Linux porting guide

Rev. 1.1 — 13 September 2021

Application note
COMPANY PUBLIC

Document information

Information	Content
Keywords	NFC, Linux, libnfc-nci
Abstract	This application note describes how to add support for a PN7160 NFC controller to a generic GNU/Linux system.

1 Front matter

Revision history

Rev	Date	Description
1.1	20210913	Security status changed into "Company public", no content change
1.0	20210825	Initial version

© NXP B.V. 2021. All rights reserved.

2 Introduction

This document provides guidelines for the integration of NXP's PN7160 NFC controllers to a generic GNU/Linux platform from software perspective, based on the Linux NFC stack. The related architecture is depicted in below Figure 1.

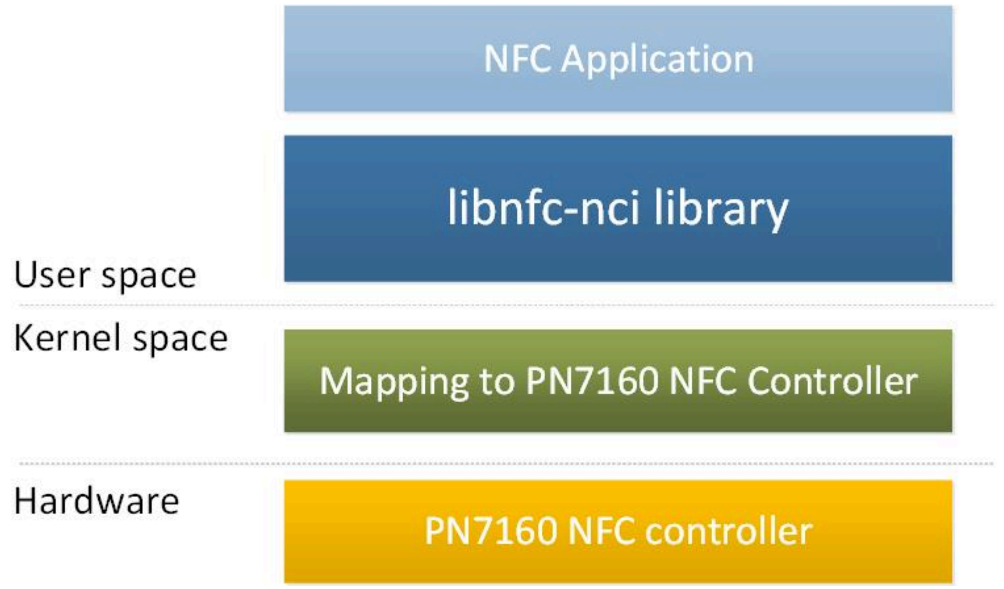


Figure 1. PN7160 Linux libnfc-nci stack overview

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

© NXP B.V. 2021. All rights reserved.

3 Release note

The present document describes the PN7160 Linux libnfc-nci stack version R1.0.

3.1 Change history

3.1.1 R1.0

First official delivery of the PN7160 Linux libnfc-nci stack.

3.2 Possible problems, known errors and restrictions

Multiple ISO15693 tags are not supported: Several tags are reported, first one can be selected but switch to the second one cannot be achieved.

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

© NXP B.V. 2021. All rights reserved.

4 Low-level access to PN7160 HW

Two possibilities are offered to allow mapping the Linux NFC stack, depicted in Figure 1, to the PN7160 NFC controller.

4.1 Kernel driver nxpnfc

The nxpnfc kernel driver can be used to communicate with the PN7160 NFC controller. Source code is available from the following repository: https://github.com/NXPNFCLinux/nxpnfc.

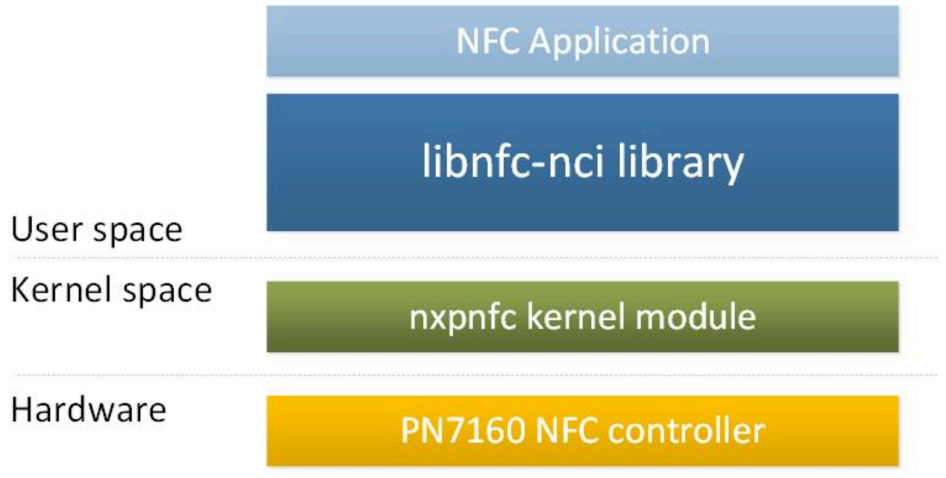


Figure 2. Linux libnfc-nci stack with nxpnfc kernel driver

4.1.1 Driver details

The nxpnfc kernel driver offers communication to the NFC controller connected over either I^2C or SPI physical interface.

When loaded to the kernel, this driver exposes the interface to the NFC controller through the device node named */dev/nxpnfc*.

This kernel driver is compatible with a broad range of NXP's NFC controllers, it explains specific NXP references can be found in the source code.

The provided source code allows building both versions of the kernel driver (I^2C and SPI) according to the kernel configuration.

4.1.2 Getting the driver

Clone the nxpnfc repository into the kernel directory, replacing existing implementation:

```
$ rm -rf drivers/nfc
$ git clone https://github.com/NXPNFCLinux/nxpnfc.git drivers/nfc
```

This will end-up with the folder *drivers/nfc* containing the following files:

- *README.md*: repository information
- *Makefile*: driver heading makefile
- *Kconfig*: driver configuration file
- *LICENSE*: driver licensing terms

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

© NXP B.V. 2021. All rights reserved.

- *i2c_devicetree.txt*: example of I²C device tree definition
- *spi_devicetree.txt*: example of SPI device tree definition
- *nfc* sub folder containing:
  - *Makefile*:
  - *common.c*: generic driver implementation
  - *common.h*: generic driver interface definition
  - *i2c.c*: I^{2}C specific driver implementation
  - *i2c.h*: I^{2}C specific driver interface definition
  - *spi.c*: SPI-specific driver implementation
  - *spi.h*: SPI-specific driver interface definition

4.1.3 Including the driver to the kernel

Including the driver to the kernel, and making it loaded during device boot, is done thanks to the device tree.

After updating the device tree definition as suggested in below examples, the platform-related device tree must be rebuilt.

4.1.3.1 I²C version

I^{2}C address (0x28 in below examples) and GPIO assignments must be adapted according to the hardware integration in the platform.

Below is an example of definition to be added to the platform device tree file (*.dts* file located for instance under *arch/arm/boot/dts* kernel subfolder for arm-based platform).

```
i2c0: i2c@ffd71000 {
    ...
    status = "ok";
    nxpnfc: nxpnfc@28 {
        compatible = "nxp,nxpnfc";
        reg = <0x28>;
        nxp,nxpnfc-irq = <&gpio26 0 0>;
        nxp,nxpnfc-ven = <&gpio26 2 0>;
        nxp,nxpnfc-fw-dwnld = <&gpio26 4 0>;
    };
};
```

4.1.3.2 SPI version

SPI handle (0 in the below example) and GPIO assignments must be adapted according to the hardware integration in the platform.

Below is an example of definition to be added to the platform device tree file (*.dts* file located for instance under *arch/arm/boot/dts* kernel subfolder for arm-based platform).

```
spi2: spi@ffd68000 {
    ...
    status = "ok";
    nxpnfc@0 {
        compatible = "nxp,nxpnfc";
        reg = <0>;
        nxp,nxpnfc-irq = <&gpio26 0 0>;
        nxp,nxpnfc-ven = <&gpio26 2 0>;
        nxp,nxpnfc-fw-dwnld = <&gpio26 4 0>;
        spi-max-frequency = <7000000>;
    };
};
```


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

© NXP B.V. 2021. All rights reserved.

4.1.4 Building the driver

Through *menuconfig* procedure include the targeted driver (I²C or SPI version) to the build, as built-in (<*>):

```
Device Drivers --->
    < > NFC I2C Slave driver for NXP-NFCC
    < > NFC SPI Slave driver for NXP-NFCC
```

Rebuilding the complete kernel, the driver will be included in the kernel image.

4.1.5 Changing access to device node

By default, r/w permission to the ***/dev/nxpnfc*** node is set to root user only. This might be an issue when running an application without root privilege.

Permissions of the device node can be changed on the platform, by instance using ***udev*** rules management For example, creating a new file named ***nxpnfc.rules*** located in ***/etc/udev/rules.d*** platform sub-directory, and containing such line declaration:

```
ACTION=="add", KERNEL=="nxpnfc", MODE="0666"
```

This will update the device node permission, to r+w to any user, during platform boot.

4.2 Alternative to nxpnfc kernel driver

In case the existing kernel offers access to GPIO and I²C or SPI resources from the user space (through ***/sys/class/gpio*** and ***/dev/i2c-****X* or***/dev/spidev****X.X* interface), an alternative to the nxpnfc kernel driver is proposed. This is managed inside the Hardware Abstraction Layer component of the libnfc-nci SW stack and selected from the ***libnfc-nxp.conf*** configuration file via "NXP_TRANSPORT" parameter (see Table 2).

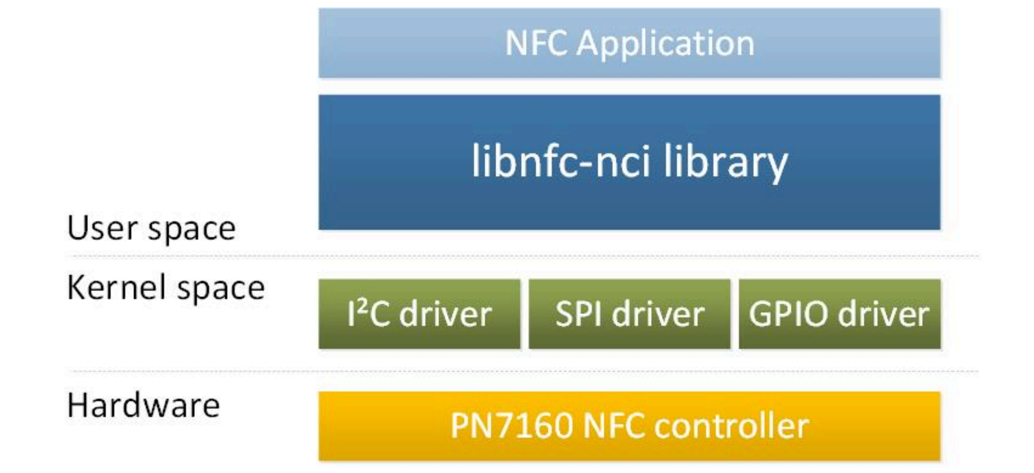


Figure 3. Linux libnfc-nci stack with alternative to nxpnfc kernel driver

When accessing the NFC Controller through ***/sys/class/gpio*** and ***/dev/i2c-****X* or ***/dev/spidev****X.X*, related rights must be insured to the NFC application (either the NFC application must be executed as root or rights must be extended to user).

© NXP B.V. 2021. All rights reserved.

The GPIO connection, as well as I^2C or SPI, to the NFC Controller are defined in *libnfc-nci/src/nfcandroid_nfc_hidlimpl/halimpl/tml/transport/NfccAltTransport.h* file and must be adapted to the targeted platform:

```
#define PIN_INT         23
#define PIN_ENABLE      24
#define PIN_FWDNLD      25
```

- {PIN_INT} defines the GPIO number the NFC Controller IRQ pin is connected to
- {PIN_ENABLE} defines the GPIO number the NFC Controller VEN pin is connected to
- {PIN_FWDNLD} defines the GPIO number the NFC Controller DWL_REQ pin is connected to

```
#define I2C_BUS         "/dev/i2c-1"
#define I2C_ADDRESS     0x28
```

- {I2C_BUS} defines the I^2C device instance the NFC Controller is connected to
- {I2C_ADDRESS} defines the NFC Controller 7 bits I^2C slave address

```
#define SPI_BUS         "/dev/spidev0.0"
```

- {SPI_BUS} defines the SPI device instance the NFC Controller is connected to

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

© NXP B.V. 2021. All rights reserved.

5 NFC library

The Linux libnfc-nci stack consists in a library running in User space. It is available from the following repository: https://github.com/NXPNFCLinux/linux_libnfc-nci

5.1 Installation instructions

5.1.1 Getting the library source code

Clone the Linux libnfc-nci stack repository:

```
$ git clone https://github.com/NXPNFCLinux/linux_libnfc-nci.git -b NCI2.0_PN7160
```

The following directory structure will be created:

```
├── conf
│   └── libnfc-nci.conf
│   └── libnfc-nxp.conf
├── doc
│   └── ...
├── src
│   └── ...
├── firmware
│   └── ...
├── demoapp
│   └── ...
├── .gitignore
├── LICENCE.txt
├── README.md
├── libnfc-nci.pc.in
├── bootstrap
├── configure.ac
└── Makefile.am
```

5.1.2 Generating the configuration script

Generate the configuration script by simply executing the bootstrap bash script:

```
$ ./bootstrap
```

This requires the *automake*, *autoconf* and *libtool* packages to be installed on the machine used for compilation (directly on the target or cross-compiling machine). This can be done using standard *apt-get install* procedure.

5.1.3 Generating the Makefile

Call the newly created configure script enabling the generation of the Makefile recipe file:

```
$ ./configure <OPTIONS>
```

Below are some of the options which might be interested:

- --enable-llcp1_3: enable support of LLCP1.3. Requires OpenSSL Cryptography and SSL/TLS Toolkit, if not set LLCP1.3 is not supported (falling back to LLCP1.2 support)
- openssldir=DIR: (optional) path to openssl installation folder (mandatory for LLCP1.3 support)
- *--enable-debug*: enable including debug symbols
- *-h*: display all available configure options

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

© NXP B.V. 2021. All rights reserved.

When --enable-llcp1_3 option is selected, configuration step will fail if openssldir path is not set. (e.g. "***./configure --enable-llcp1_3 openssldir=/opt/openssl***")

5.1.4 Building the source

Using the *Makefile* recipe file, building the library and the example application is done with the simple make command:

```
$ make
```

5.1.5 Installing the library

The generated library binaries and header files can be installed on the target using *make install* command.

Depending on the target directories, installation may require the use of root privileges, generally granted by *su* or *sudo*:

```
$ sudo make install
```

It installs the libnfc-nci-linux library to */usr/local/lib* target directory. This path must be added to *LD_LIBRARY_PATH* environment variable for proper reference to the library during linking/execution of related application.

It also installs the configuration files (refer to chapter Section 5.3) to */usr/local/etc* and the library header files to */usr/local/include*.

5.2 Library APIs

For detailed information about libnfc-nci library API, please refer to the dedicated document *Linux_NFC_API_Guide.html* inside *doc* sub-folder of the stack delivery (refer to chapter Section 5.1.1).

5.3 Configuration files

Two files allow configuring the libnfc-nci library at runtime: *libnfc-nci.conf* and *libnfc-nxpconf*. There are defining tags which are impacting library behavior. The value of the tags depends on the targeted platform. For more details, refer to the examples given in *conf* sub-folder of the stack delivery (see chapter Section 5.1.1).

These files are loaded by the library, from */usr/local/etc* directory of the target, during the initialization phase. Refer to chapter Section 5.1.5 for installation procedure, the files can also be manually copied to the target */usr/local/etc* directory.

Pay attention that the configuration files provided as part of the library relates to the NFC controller dev boards. Some parameters must have to be adapted according to the target integration.

Below is the description of the different useful tags in the configuration files (refer to the example conf files for detailed information about the tag values).

Table 1. Tag list of libnfc-nci.conf file

Tag	Description
APPL_TRACE_LEVEL	Log levels for libnfc-nci. Recommended value for debugging is 0xFF

© NXP B.V. 2021. All rights reserved.

Table 1. Tag list of libnfc-nci.conf file...*continued*

Tag	Description
PROTOCOL_TRACE_LEVEL	Log levels for libnfc-nci. Recommended value for debugging is 0xFF
HOST_LISTEN_ENABLE	Configuration force HOST listen feature
POLLING_TECH_MASK	Configuration of the polling technologies
NFA_DM_DISC_DURATION_POLL	Configuration of the discovery loop TOTAL DURATION (in milliseconds)
P2P_LISTEN_TECH_MASK	Configuration of listen technologies for P2P

Table 2. Tag list of libnfc-nxp.conf file

Tag	Description
NXPLOG_EXTNS_LOGLEVEL	Configure level of EXTNS logs Recommended value for debug is 0x03
NXPLOG_NCIX_LOGLEVEL	Set level of NCIX logs Recommended value for debug is 0x03
NXPLOG_NCIR_LOGLEVEL	Set level of NCIR logs Recommended value for debug is 0x03
NXPLOG_FWDNLD_LOGLEVEL	Set level of FWDNLD logs Recommended value for debug is 0x03
NXPLOG_TML_LOGLEVEL	Set level of FWDNLD logs Recommended value for debug is 0x03
NXP_ACT_PROP_EXTN	Set NXP's NFC Controller proprietary features
NXP_NFC_PROFILE_EXTN	Set discovery profile
NXP_CORE_STANDBY	Set NFC Controller standby mode
NXP_AGC_DEBUG_ENABLE	Set the dynamic RSSI feature
NXP_TRANSPORT	Defines the transport configuration 0x00 means kernel driver, 0x02 is Altenative I^2C, 0x03 Alternative SPI and other values are RFU
NXP_NFC_DEV_NODE	Set the device node when kernel driver transport configuration is used
NXP_I2C_FRAGMENTATION_ENABLED	Set the I2c fragmentation feature
NXP_NFC_FW_PATH	Defines path from which the library shall load the NFC Controller firmware
NXP_NFC_FW_NAME	Defines NFC Controller firmware library name to be loaded
MIFARE_READER_ENABLE	Set the support of the reader for MIFARE Classic
NXP_NFC_PROPRIETARY_CFG	Defines the proprietary protocols ID used in discovery loop
NXP_SYS_CLK_SRC_SEL	Configure the clock source of the NFC Controller
NXP_SYS_CLK_FREQ_SEL	Set the clock frequency in case of PLL clock source

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

© NXP B.V. 2021. All rights reserved.

Table 2. Tag list of libnfc-nxp.conf file*...continued*

Tag	Description
NXP_SYS_CLOCK_TO_CFG	Set clock request acknowledgment time value in case of PLL clock source
NXP_EXT_TVDD_CFG	Set TVDD configuration used
NXP_EXT_TVDD_CFG_1	Configure TxLDO when CFG1 is used
NXP_EXT_TVDD_CFG_2	Configure TxLDO when CFG2 is used
NXP_RF_CONF_BLK_x	Set platform-specific RF configuration
NXP_CORE_CONF_EXTN	Configure proprietary settings of the NFC Controller
NXP_CORE_CONF	Configure standardized settings of the NFC Controller
NXP_CORE_MFCKEY_SETTING	Proprietary configuration for Key storage
NFA_MAX_EE_SUPPORTED	Set the maximum number of Execution Environments supported

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

© NXP B.V. 2021. All rights reserved.

6 Example application

6.1 Application details

The Linux libnfc-nci stack offers an application example demonstrating use of the library to run NFC features. It is available as part of the stack delivery (refer to chapter Section 5.1 for installation instructions). Source code is located in *demoapp* sub-folder of the libnfc-nci stack directory.

The purpose of this application is to demonstrate NFC features offers by the libnfc-nci library and provides code example of the library API.

It is built together the libnfc-nci library, following procedure depicted in chapter Section 5.1.4.

6.2 Using the application

The application must be started with parameters:

```
$ ./nfcDemoApp <OPTIONS>
```

You can get the parameters details by launching the application help menu:

```
$ ./nfcDemoApp --help
```

Figure 4. Linux demo application commands

The demo application offers 3 modes of operation:

- **Polling**: continuously waiting for a remote NFC device (tag or peer device) and displays related information
- **Tag writing**: allows writing NDEF content to an NFC tag
- **Tag emulation**: allows sharing NDEF content to an NFC reader device
- **Device push**: allows pushing NDEF content to a remote NFC peer device

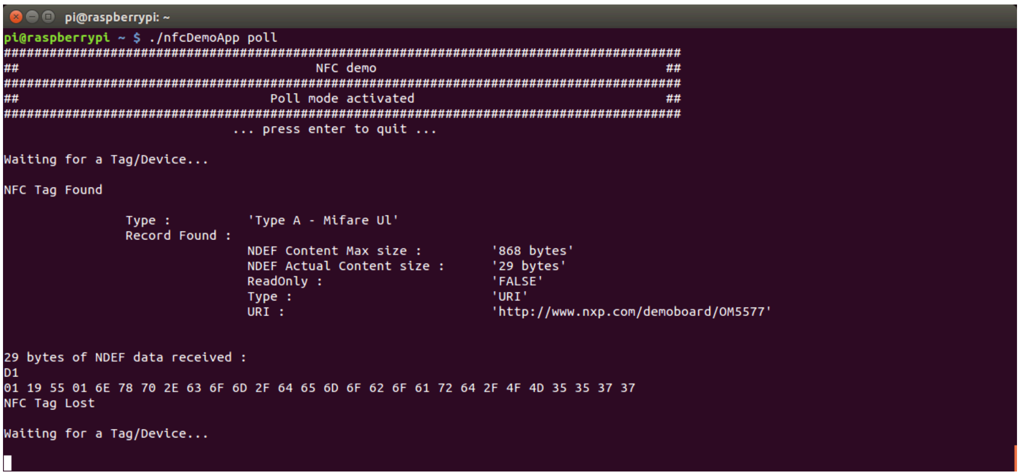
6.2.1 Run Polling mode

When in this mode, the application displays information of any discovered NFC tags or remote NFC device. It is reached starting the application with “poll” parameter:

```
$ ./nfcDemoApp poll
```


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

© NXP B.V. 2021. All rights reserved.



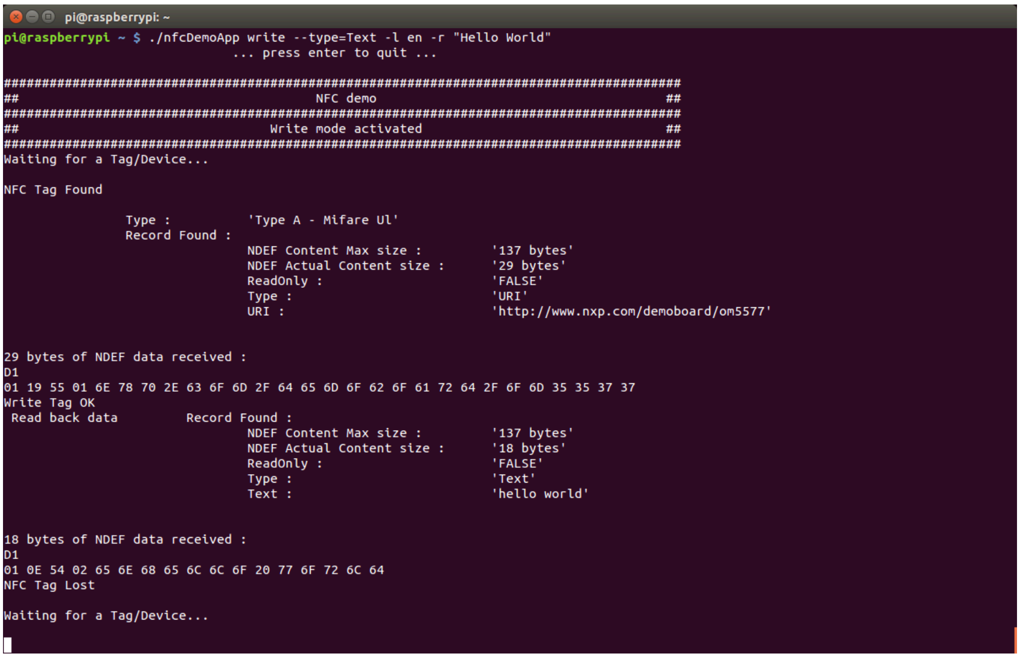
```
pi@raspberrypi ~ $ ./nfcDemoApp poll
##########################################################################################
##                                        NFC demo                                      ##
##########################################################################################
##                                  Poll mode activated                                 ##
##########################################################################################
                              ... press enter to quit ...

Waiting for a Tag/Device...

NFC Tag Found

                Type :          'Type A - Mifare Ul'
                Record Found :
                                NDEF Content Max size :         '868 bytes'
                                NDEF Actual Content size :      '29 bytes'
                                ReadOnly :                      'FALSE'
                                Type :                          'URI'
                                URI :                           'http://www.nxp.com/demoboard/OM5577'


29 bytes of NDEF data received :
D1
01 19 55 01 6E 78 70 2E 63 6F 6D 2F 64 65 6D 6F 62 6F 61 72 64 2F 4F 4D 35 35 37 37
NFC Tag Lost

Waiting for a Tag/Device...
```

Figure 5. Linux demo application polling mode

6.2.2 Tag writing mode

This mode allows writing data to an NFC tag. It is reached using “write” parameter:

```
$ ./nfcDemoApp write <OPTIONS>
```



```
pi@raspberrypi ~ $ ./nfcDemoApp write --type=Text -l en -r "Hello World"
                              ... press enter to quit ...

##########################################################################################
##                                        NFC demo                                      ##
##########################################################################################
##                                  Write mode activated                                ##
##########################################################################################
Waiting for a Tag/Device...

NFC Tag Found

                Type :          'Type A - Mifare Ul'
                Record Found :
                                NDEF Content Max size :         '137 bytes'
                                NDEF Actual Content size :      '29 bytes'
                                ReadOnly :                      'FALSE'
                                Type :                          'URI'
                                URI :                           'http://www.nxp.com/demoboard/om5577'


29 bytes of NDEF data received :
D1
01 19 55 01 6E 78 70 2E 63 6F 6D 2F 64 65 6D 6F 62 6F 61 72 64 2F 6F 6D 35 35 37 37
Write Tag OK
 Read back data         Record Found :
                                NDEF Content Max size :         '137 bytes'
                                NDEF Actual Content size :      '18 bytes'
                                ReadOnly :                      'FALSE'
                                Type :                          'Text'
                                Text :                          'hello world'


18 bytes of NDEF data received :
D1
01 0E 54 02 65 6E 68 65 6C 6C 6F 20 77 6F 72 6C 64
NFC Tag Lost

Waiting for a Tag/Device...
```

Figure 6. Linux demo application tag writing mode

You can get more information about the message format using “-h” or “--help” parameter:

```
$ ./nfcDemoApp write --help
```


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

© NXP B.V. 2021. All rights reserved.

6.2.3 Tag emulation mode

This mode allows emulating an NFC tag (NFC Forum T4T) to send data to a remote NFC reader (e.g. an NFC phone). It is reached using “share” parameter:

```
$ ./nfcDemoApp share <OPTIONS>
```

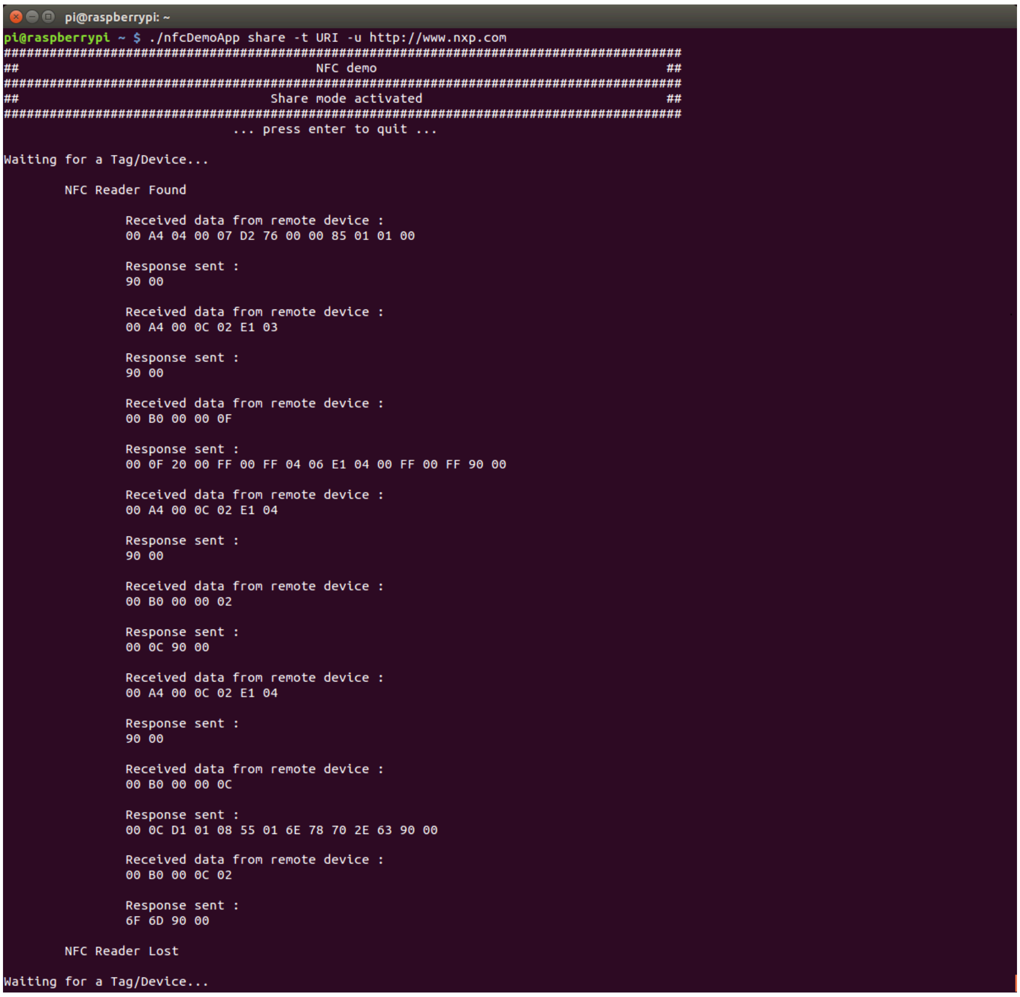


Figure 7. Linux demo application Tag emulation mode

You can get more information about the message format using “-h” or “--help” parameter:

```
$ ./nfcDemoApp share --help
```

6.2.4 Device push mode

This mode allows pushing data to a remote NFC device (e.g. an NFC phone). It is reached using “push” parameter:

```
$ ./nfcDemoApp push <OPTIONS>
```


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

© NXP B.V. 2021. All rights reserved.

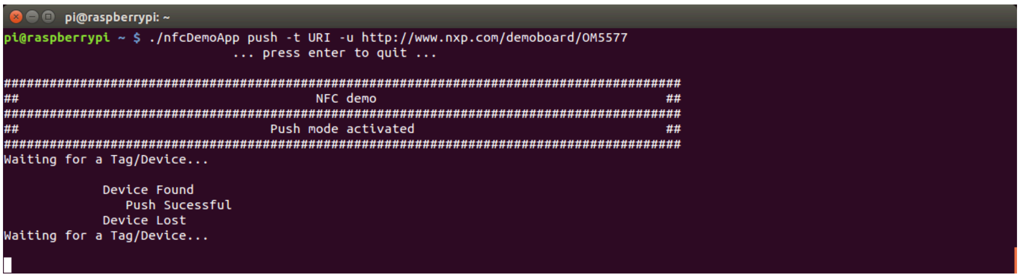


Figure 8. Linux demo application device push mode

You can get more information about the message format using “-h” or “--help” parameter:

```
$ ./nfcDemoApp push --help
```


© NXP B.V. 2021. All rights reserved.

7 Additional specific examples

Additional examples are available demonstrating specific NFC functionalities of the Linux libnfc-nci library. They can be retrieved by cloning the https://github.com/NXPNFCLinux/linux_libnfc-nci_examples:

```
$ git clone https://github.com/NXPNFCLinux/linux_libnfc-nci_examples
```

Each example can be individually built using the dedicated *Makefile* recipe file with the simple "make" command. Since depending on the Linux libnfc-nci library, it requires the library file and configuration files are installed on the target (see Section 5.1.5).

© NXP B.V. 2021. All rights reserved.

8 NFC Factory Test application

8.1 Application details

To ease the characterization of the NFC integration in the Linux device, the NFC Factory Test application is offered. It allows setting the NFC controller into either:

- Constant RF emission mode (no modulation)
- Functional mode (card detection)
- PRBS (Pseudo Random Binary Sequence) mode (continuous modulation)
- Standby mode (for power consumption measurement)

Additionally, the application allows to:

- Dump all RF settings values
- Set RF settings
- Get NCI parameters value
- Set NCI parameters value
- Get proprietary parameters value
- Set proprietary parameters value

The source code is available from the following repository: https://github.com/NXPNFCLinux/linux_NfcFactoryTestApp.

This application does not run on top of the libnfc-nci SW stack but instead directly access the NFC Controller to send the appropriate NCI commands allowing to set it into the expected mode.

8.2 Building the application

Clone the NFC Factory Test application repository:

```
$ git clone https://github.com/NXPNFCLinux/linux_NfcFactoryTestApp.git
```

Using the *Makefile* recipe file, build the application with the “make” command:

```
$ make
```

This will generate the application based on the nxpnfc kernel driver for the communication to the NFC controller (see Section 4.1). If the integration is based on the alternative option (refer to Section 4.2), the application must be built using the “alt-i2c” parameter (if NFC Controller is connected over I²C interface) or “alt-spi” parameter (if NFC Controller is connected over SPI interface):

- For Alternative I²C:

  ```
  $ make alt-i2c
  ```

  I²C and GPIO connection details being defined in *tml_alt-i2c.c* file:

  ```
  #define I2C_BUS         "/dev/i2c-1"
  #define I2C_ADDRESS     0x28

  #define PIN_INT         23
  #define PIN_ENABLE      24
  ```


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

© NXP B.V. 2021. All rights reserved.

- For Alternative SPI:

```
$ make alt-spi
```

SPI and GPIO connection details being defined in *tml_alt-spi.c* file:

```
#define SPI_BUS         "/dev/spidev0.0"
#define SPI_MODE        SPI_MODE_0;
#define SPI_BITS        8;
#define SPI_SPEED       5000000

#define PIN_INT         23
#define PIN_ENABLE      24
```

Run the application, pay attention that the low-level access rights to the device node are granted (***/dev/nxpnfc***,***/dev/i2c-****X* or ***/dev/spidev****X.X*):

```
$ ./NfcFactoryTestApp
```

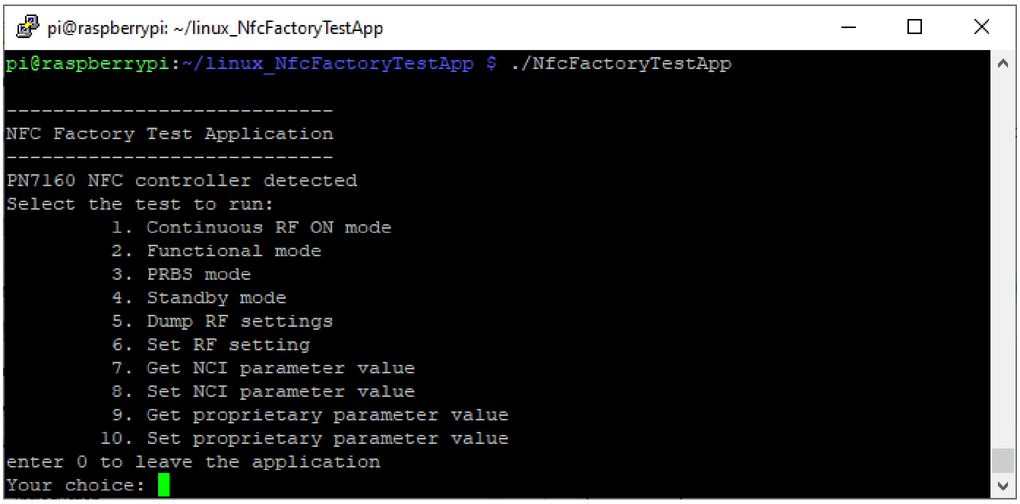


Figure 9. NFC Factory Test application

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

© NXP B.V. 2021. All rights reserved.

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 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 or documented vulnerabilities. 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.

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.

© NXP B.V. 2021. All rights reserved.

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.

© NXP B.V. 2021. All rights reserved.

Tables

© NXP B.V. 2021. All rights reserved.

Figures

© NXP B.V. 2021. All rights reserved.

Contents

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

© NXP B.V. 2021. **All rights reserved.**

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 13 September 2021
Document identifier: AN13287