AN11697

PN71xx Linux Software Stack Integration Guidelines

Rev. 2.3 — 12 June 2017
335023

Application note
COMPANY PUBLIC

Document information

Info	Content
Keywords	NFC, Linux, Libnfc-nci
Abstract	This note describes how to add support for a PN7120 or PN7150 NFC Controller to a generic GNU/Linux system

Revision history

Rev	Date	Description
2.3	20170612	• Updated to R2.2 software stack version • Added description of NFC Factory Test application
2.2	20160704	• Updated to R2.1 software stack version • Added description of configuration files
2.1	20160509	Added support for PN7150 NFCC IC
2.0	20160223	• Updated to R2.0 software stack version • Updated Section 7.3 Licenses
1.1	20150824	Updated to R1.0 software stack version
1.0	20150601	First released version
0.1	20150507	Creation of the document

Contact information

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

1. Introduction

This document provides guidelines for the integration of NXP's PN7120 and PN7150 NFC Controllers to a generic GNU/Linux platform from software perspective, based on the Linux NFC stack. The related architecture is depicted in below Fig 1.

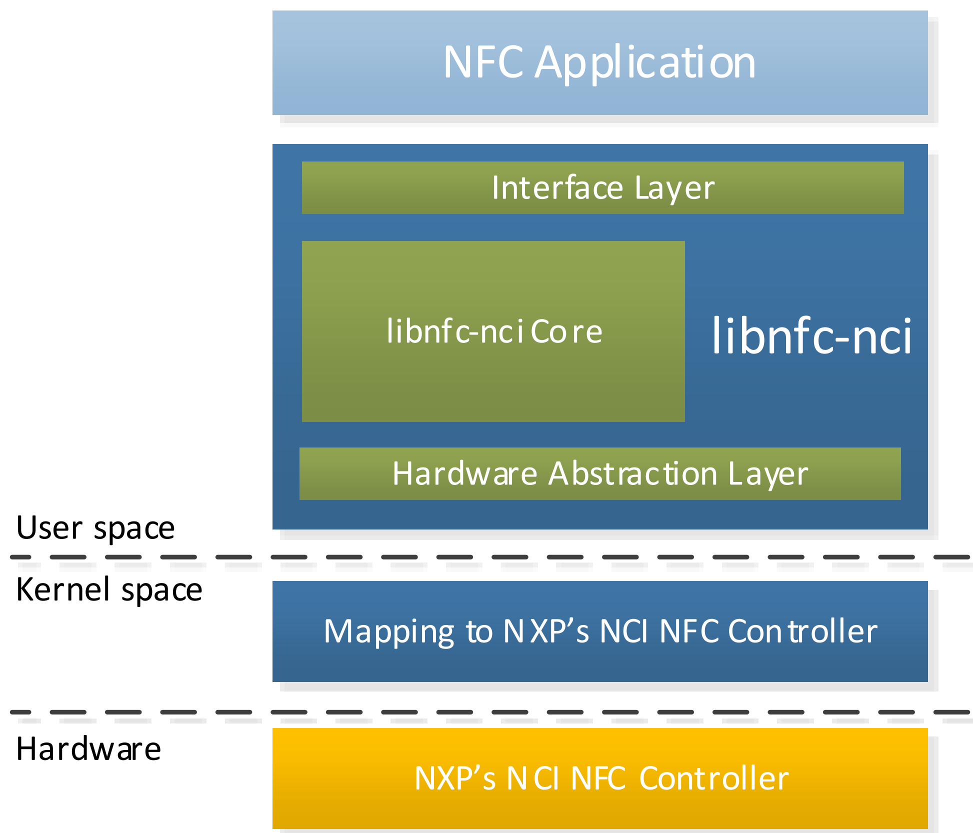

Fig 1. Linux libnfc-nci stack overview

2. Release note

The present document describes the Linux libnfc-nci stack version R2.2.

2.1 Change history

2.1.1 R2.2

- Adding support for alternative to pn5xx_i2c kernel driver
- Adding support for lpcusbsio based device
- Updated demo application with MIFARE cards additional examples
- Fixed issue with presence check mechanism with respect to MIFARE classic scenario
- Fixed SNEP server creation issue on x64/x86 platforms
- Fixed conflict when using latest version of openssl
- Fixed syntax error in Self-Test implementation
- Fixed issue with I2C fragmentation
- Updates of PN7150 RF settings configuration

2.1.2 R2.1

- Adding support for PN7150 NFCC IC
- Fixed issue of RF block write fail
- Fixed issue of HCE not working when polling tech mask is null
- Fixed issue of listen mode for HCE always enabled when NXP_NFC_NATIVE_ENABLE_HCE=false
- Updated PN7150 configuration file in the scope of RF performance optimization of OM5578 demo kit

2.1.3 R2.0

- Implementation of LLCP1.3 (enable/disable through configuration)
- Implementation of P2P Connectionless data transfer for LLCP1.2 and LLCP1.3
- Fixed Read NDEF return status on failure
- Fixed P2P RF issue during multiple transfer in LLCP1.3
- Fixed Segmentation fault observed during secured P2P transfer
- Fixed thread creation issue discovered in specific case of endurance testing with remote tag set at the limit of detection
- Fixed buffer allocation issue discovered in specific case of endurance testing with remote tag set at the limit of detection

2.1.4 R1.0

- Fixed SNEP Connect Error on PC x64 platform
- Fixed issue of failing initialization due to DWL_GPIO not connected or not defined

- Fixed error during receive of BT CHO message
- Fixed error during receive of WIFI CHO message
- Fixed issue of Handover select API returning success for corrupted payload
- Fixed issue of HCE data receive call back not being invoked in case of receiving check NDEF frame
- Fixed error during multiprotocol card reading
- Fixed status error when push message failed
- Fixed segmentation fault in case of URI NDEF record with invalid/RFU prefixes
- Fixed MIFARE Classic buffer de-allocation during write
- Fixed RF Stuck issue during P2P or tag write

2.1.5 R0.4

First official delivery of the Linux libnfc-nci stack.

2.2 Possible problems, known errors and restrictions

LLCP1.3 support requires OpenSSL Cryptography and SSL/TLS Toolkit (version 1.0.1j or later)

3. Low level access to PN71xx HW

Several different possibilities are offered to allow mapping the Linux NFC stack, depicted in Fig 1, to the PN71xx NFC Controller.

3.1 Kernel driver pn5xx_i2c

PN5xx I2C kernel mode driver can be used to communicate with the PN71xx NFC Controller. Source code is available from the following repository: http://www.github.com/NXPNFCLinux/nxp-pn5xx .

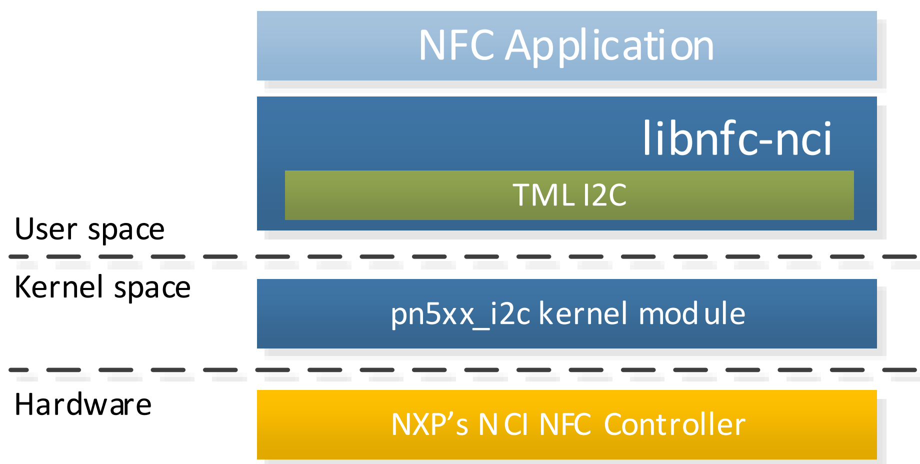

Fig 2. Linux libnfc-nci stack with pn5xx_i2c kernel driver

3.1.1 Driver details

The PN5xx I2C driver offers communication to the NFC Controller connected over I2C physical interface. This is insured through the device node named */dev/pn544*. This driver is compatible with a large range of NXP's NFC Controllers (e.g. PN544).

3.1.2 Installation instructions

The following instructions assume the driver being installed under the *drivers/misc* kernel source sub-folder. Below instructions may have to be adapted accordingly in case another path is chosen for the driver installation.

3.1.2.1 Getting the driver

Clone the nxp-pn5xx repository into the kernel directory:

```
$ cd drivers/misc
$ git clone https://github.com/NXPNFCLinux/nxp-pn5xx.git
```

This will create the sub-folder *nxp-pn5xx* containing the following files:

- *pn5xx_i2c.c*: driver implementation
- *pn5xx_i2c.h*: driver interface definition
- *README.md*: repository comments
- *Makefile*: driver related makefile
- *Kconfig*: driver related config file
- *LICENSE*: driver licensing terms
- *sample_devicetree.txt*: example of device tree definition

3.1.2.2 Including the driver to the kernel

Include the driver to the compilation by adding below line to the heading makefile (*drivers/misc/Makefile*).

```
obj-y          += nxp-pn5xx/
```

Include the driver config by adding below line to the heading configuration file (*drivers/misc/Kconfig*).

```
source "drivers/misc/nxp-pn5xx/Kconfig"
```

3.1.2.3 Creating the device node

Two methods are supported for the creation of the */dev/pn544* device node: device tree and platform data. Any of the two methods can be used, but of course the I2C address (0x28 in the below examples) and GPIO assignments must be adapted to the hardware integration in the platform.

i. Device tree

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 sub-folder for ARM based platform).

```
&i2c{
  status = "okay";
  pn547: pn547@28 {
    compatible = "nxp,pn547";
    reg = <0x28>;
    clock-frequency = <400000>;
    interrupt-gpios = <&gpio2 17 0>;
    enable-gpios = <&gpio4 21 0>;
  };
};
```

ii. Platform data

Below is an example of definition to be added to the platform definition file. The structure *pn544_i2c_platform_data* being defined in the driver interface header file, *pn5xx_i2c.h* must be included in the platform definition file, and *pn5xx_i2c.h* file must be copied to *include/linux* kernel source sub-folder.

```
static struct pn544_i2c_platform_data nfc_pdata = {
  .irq_gpio = GPIO_TO_PIN(1,29),
  .ven_gpio = GPIO_TO_PIN(0,30),
  .firm_gpio = GPIO_UNUSED
  .clkreq_gpio = GPIO_UNUSED
};

static struct i2c_board_info __initdata nfc_board_info[] = {
  {
    I2C_BOARD_INFO("pn547", 0x28),
    .platform_data = &nfc_pdata,
  },
};
```

Then the declared ***nfc_board_info*** structure must be added to the platform using dedicated procedure (platform specific).

3.1.2.4 Building the driver

Through *menuconfig* procedure include the driver to the build, as built-in (<*>) or modularizes features (<M>):

```
Device Drivers --->
     Misc devices --->
          < > NXP PN5XX based driver
```

If <M> option is selected, build the driver and install the generated *pn5xx_i2c.ko* module. Otherwise if built-in, build the complete kernel, the driver will be included in the kernel.

If the device tree method was used in previous step, build the platform related device tree and install generated dtb file.

3.1.2.5 Changing access to device node

By default, r/w permission to the ***/dev/pn544*** 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 ***pn5xx_i2c.rules*** located in ***/etc/udev/rules.d*** platform sub-directory, and containing such line declaration:

```
ACTION=="add", KERNEL=="pn544", MODE="0666"
```

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

3.2 Alternative to pn5xx_i2c kernel driver

In case the existing kernel offers access to GPIO and I2C resources from the user space (through ***/sys/class/gpio*** and ***/dev/i2c-dev*** interface), an alternative to the pn5xx_i2c kernel driver is proposed. This is managed inside the Hardware Abstraction Layer component of the libnfc-nci SW stack and selected at the "Makefile generation" step (see 4.2.2) using "--enable-alt" option.

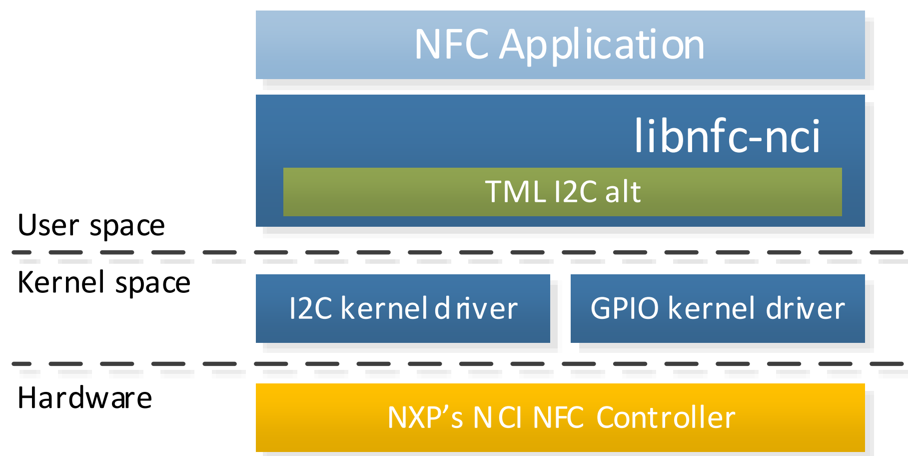

Fig 3. Linux libnfc-nci stack with alternative to pn5xx_i2c kernel driver

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

The I2C and GPIO connection to the NFC Controller is depicted in *phTmlNfc_alt.h* file and must be adapted to the targeted platform:

```
#define I2C_BUS          "/dev/i2c-1"
#define I2C_ADDRESS      0x28
#define PIN_INT          23
#define PIN_ENABLE       24
```

- {I2C_BUS} defines the I2C dev instance the NFC Controller is connected to
- {I2C_ADDRESS} defines the NFC Controller 7 bits I2C slave address
- {PIN_INT} defines the GPIO number the NFC Controller IRQ pin is connected to
- {I2C_ENABLE} defines the GPIO number the NFC Controller VEN pin is connected to

3.3 LPCUSBSIO based device

For USB devices implementing PN7150 with HID interface expose via LPCUSBSIO protocol, the libnfc-nci stack implement the support of this communication. This is managed inside the Hardware Abstraction Layer component of the libnfc-nci SW stack and selected at the “Makefile generation” step (see 4.2.2) using “--enable-lpcusbsio” option.

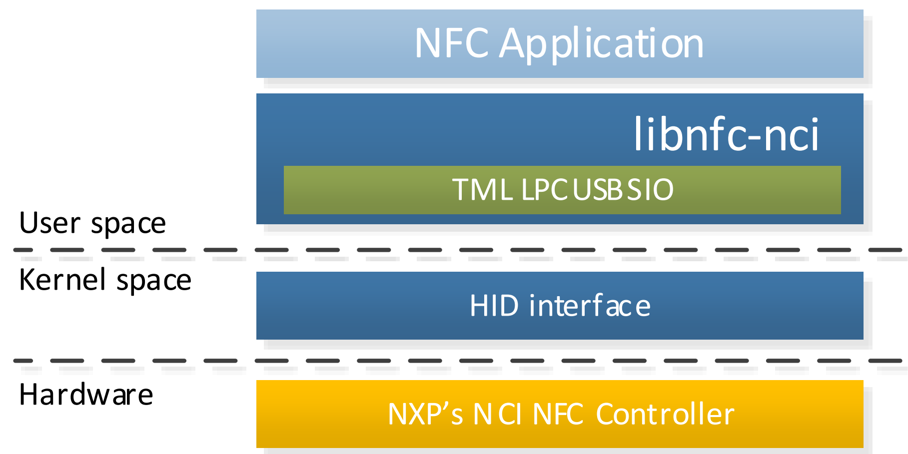

Fig 4. Linux libnfc-nci stack with lpcusbsio HID communication

4. 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

4.1 Library details

The library is comprised of 3 layers:

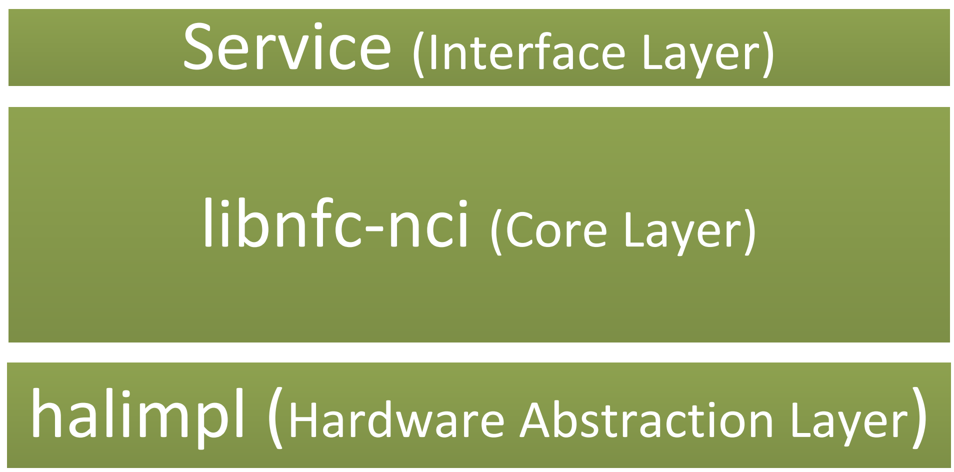

Fig 5. Linux libnfc-nci library overview

- The Interface layer expose the library API
- The Core layer implement NFC features (NCI, NDEF, LLCP and SNEP protocols, Tag Operations, Host Card Emulation…)
- The Hardware Abstraction Layer provides connection to the kernel driver as well as basic functionalities like self-test or firmware update

4.2 Installation instructions

4.2.1 Getting the library

Clone the Linux libnfc-nci stack repository:

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

The following directory structure will be created:

```
├── conf
│   └── PN7120
│       └── libnfc-brcm.conf
│       └── libnfc-nxp.conf
│   └── PN7150
│       └── libnfc-brcm.conf
│       └── libnfc-nxp.conf
├── demoapp
│       └── ...
├── doc
│   └── Linux_NFC_API_Guide
│       └── ...
│   └── Linux_NFC_API_Guide.html
├── src
│   └── halimpl
│       └── ...
│   └── include
│       └── ...
│   └── libnfc-nci
│       └── ...
│   └── service
│       └── ...
├── bootstrap
├── configure.ac
└── Makefile.am
```

4.2.1 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.

4.2.2 Generating the Makefile

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

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

Here are some options one might be interested in when cross-compiling:

- --enable-pn7120 or enable-pn7150: enable NFC Controller type to PN7120 or PN7150. Default NFC Controller selection is PN7120.

- --enable-i2c, --enable-alt or --enable-lpcusbsio: define the way to communicate with the NFC Controller (see chapter 3). Default is pn5xx_i2c kernel driver.
- --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)
- --*host*=HOST: cross-compile to build programs to run on HOST.
- --*prefix*=PREFIX: install files in PREFIX. PREFIX is generally the path to either the target's or cross-compilation toolchain's /usr/directory.
- --*sysconfdir*=DIR: install configuration files in DIR.
- -*h*: display all available command-line options.

Pay attention to properly generate the Makefile according to the targeted NFC Controller IC using appropriate "*–enable-xxx"* option of the configure script. Wrong option may lead to unexpected behavior during code execution.

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***")

4.2.3 Building the source

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

```
$ make
```

4.2.4 Installing the library

The generated library 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*:

```
# make install
```

This will install 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 application).

This will also install the configuration files (refer to chapter 4.4) to the directory defined according to *--prefix*.

The libnfc-nci library expects to find the conf files in the */etc* target directory. Pay attention to set properly the path of installation during makefile generation procedure (refer to chapter 4.2.2). For instance, if the library is built on the target, configure script must have been run with option *--sysconfdir=/etc*.

4.3 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 4.2.1).

4.4 Configuration Files

Two files allow configuring the libnfc-nci library at runtime: *libnfc-brcm.conf* and *libnfc-nxp.conf*. There are defining tags which are impacting library behavior. The value of the tags depends on the NFC Controller IC and the targeted platform. For more details, refer to the examples given in *conf* sub-folder of the stack delivery (see chapter 4.2.1).

These files are loaded by the library, from */etc* directory of the target, during the initialization phase. Refer to chapter 4.2.4 for installation procedure, the files can also be manually copied to the target */etc* directory.

> Pay attention that the configuration files provided as part of the library relates to the NFC Controller demo boards. These files must be adapted according to the aimed integration.

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

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

Tag	Description
APPL_TRACE_LEVEL	Log levels for libnfc-nci. Recommended value for debugging is 0xFF.
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.
P2P_LISTEN_TECH_MASK	Configuration of listen technologies for P2P.
NFA_MAX_EE_SUPPORTED	Set the maximum number of Execution Environments supported.

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.

Tag	Description
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.
NXPLOG_NCIHAL_LOGLEVEL	Set level of NCIHAL logs. Recommended value for debug is 0x03.
MIFARE_READER_ENABLE	Set the support of MIFARE Classic Reader.
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.
NXP_SYS_CLOCK_TO_CFG	Set clock request acknowledgment time value in case of PLL clock source.
NXP_ACT_PROP_EXTN	Set NXP's NFC Controller proprietary features.
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_NFC_PROFILE_EXTN	Set discovery profile.
NXP_CORE_STANDBY	Set NFC Controller standby mode.
NXP_RF_CONF_BLK_x	Set platform specific RF configuration.
NXP_I2C_FRAGMENTATION_ENABLED	Set the I2c fragmentation feature.
NXP_CORE_CONF_EXTN	Configure proprietary parts of the NFC Controller.
NXP_CORE_CONF	Configure standardized parts of the NFC Controller.
NXP_CORE_MFCKEY_SETTING	Proprietary configuration for MIFARE Key storage.

5. Example application

5.1 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 4.2 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 4.2.3.

5.2 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
```

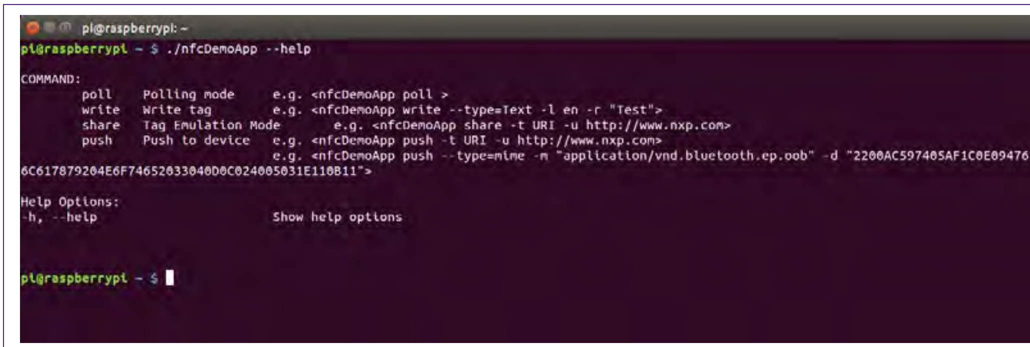

Fig 6. Linux demo application commands

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 a NFC tag
- **Tag emulation**: allows sharing NDEF content to a NFC reader device
- **Device push**: allows pushing NDEF content to a remote NFC peer device

5.2.1 Run Polling mode

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

```
$ ./nfcDemoApp poll
```

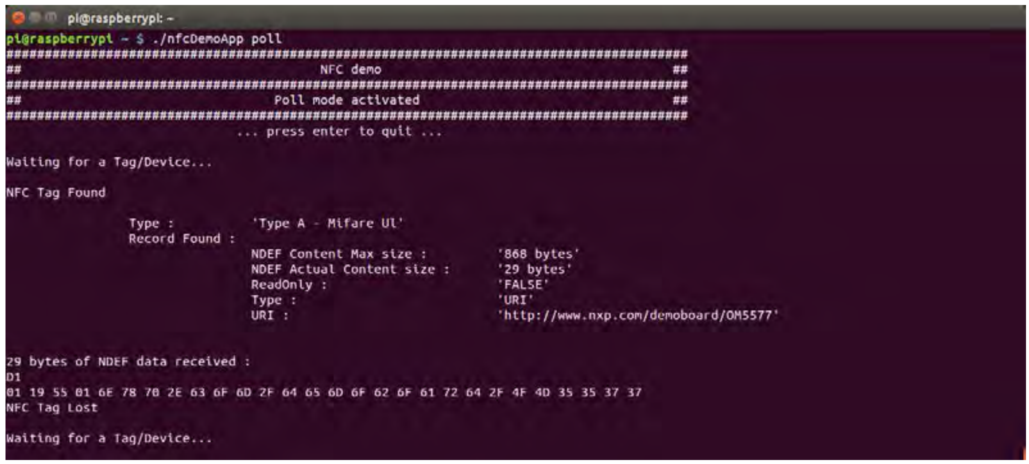

Fig 7. Linux demo application polling mode

5.2.2 Tag writing mode

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

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

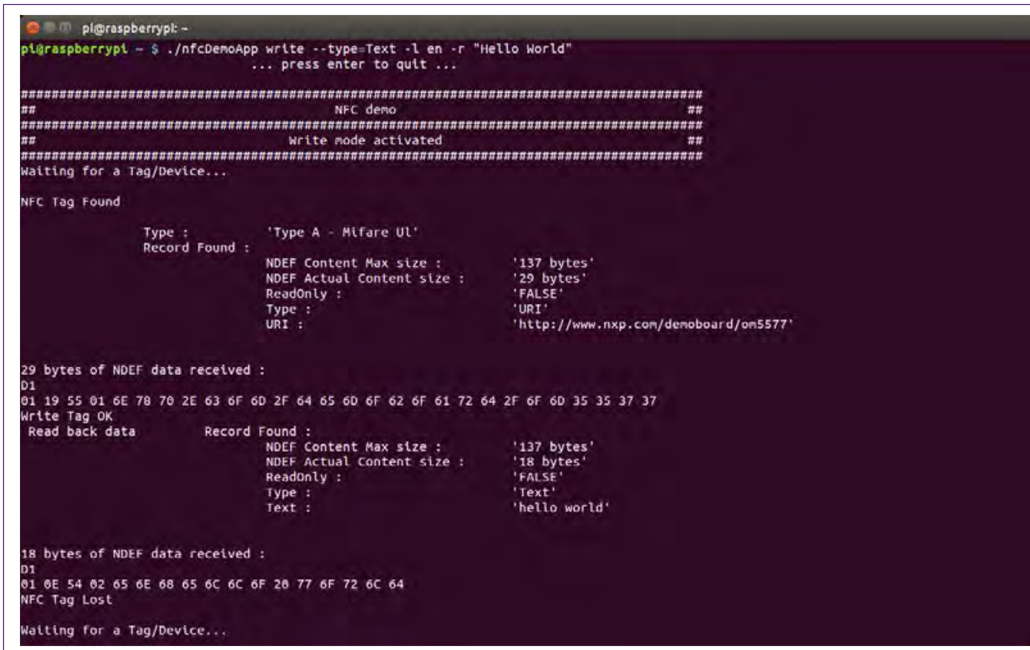

Fig 8. Linux demo application tag writing mode

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

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

5.2.3 Tag emulation mode

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

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

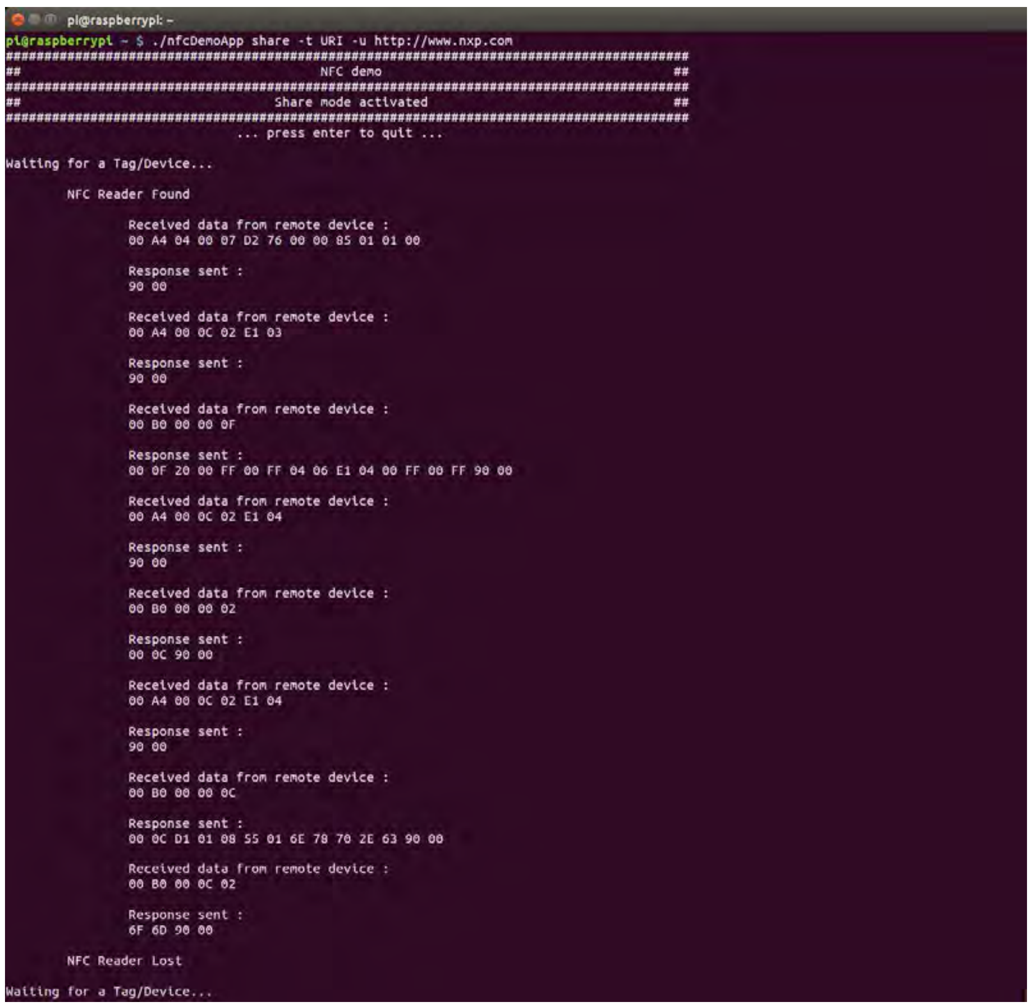

Fig 9. Linux demo application Tag emulation mode

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

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

5.2.4 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>
```

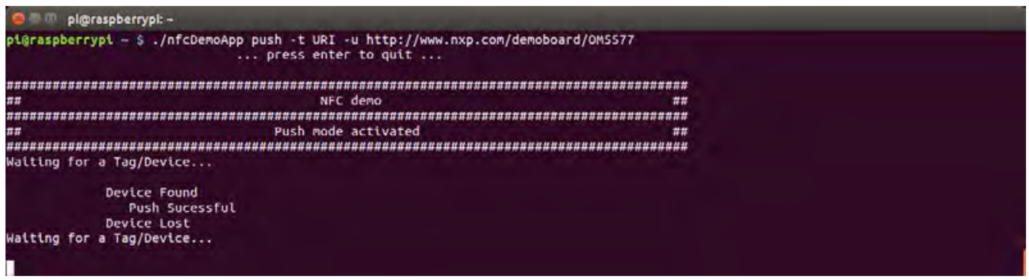

Fig 10. Linux demo application device push mode

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

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

6. NFC Factory Test application

6.1 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)
- or PRBS (Pseudo Random Binary Sequence) mode (continuous modulation)
- Standby mode

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 rather directly access the NFC Controller to send the appropriate NCI commands allowing to set it into the expected mode.

6.2 Building the application

Clone the nxp-pn5xx repository into the kernel directory:

```
$ 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 pn5xx_i2c kernel driver for the communication to the NFC Controller (see 3.1). If the integration is based on the alternative option (refer to 3.2), the application must be build using the “alt” parameter:

```
$ make alt
```

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

```
#define I2C_BUS         "/dev/i2c-1"
#define I2C_ADDRESS     0x28
#define PIN_INT         23
#define PIN_ENABLE      24
```

Run the application (pay attention that the low level access rights are given) :

```
$ ./NfcFactoryTestApp
```

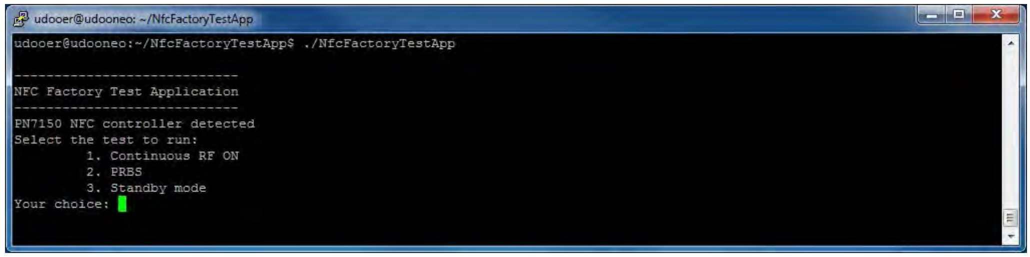

Fig 11. NFC Factory Test application

7. Legal information

7.1 Definitions

Draft — The document is a draft version only. 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 herein and shall have no liability for the consequences of use of such information.

7.2 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.

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.

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.

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.

7.3 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.

7.4 Trademarks

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

MIFARE — is a trademark of NXP B.V.

8. List of figures

9. Contents

Date of release: 12 June 2017
335023

Document identifier: AN11697