AN11480

Quick Start Up Guide for EXPLORE NFC working with Raspberry Pi

Rev. 1.0 — 17 December 2013
279710

Application note
COMPANY PUBLIC

Document information

Info	Content
Keywords	PN512; Raspberry Pi, NFC, P2P, Card Emulation, MIFARE, ISO/IEC 14443, EXPLORE-NFC
Abstract	This document gives a description on how to get started with the provided software packages in conjunction with the EXPLORE-NFC board and the Raspberry Pi Model B. It provides a step by step guide to the installation procedure of the software and the hardware.

Revision history

Rev	Date	Description
1.0	20131217	First release

Contact information

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

For sales office addresses, please send an email to: salesaddresses@nxp.com

1. Introduction

This document gives a description on how to get started with the provided software packages in conjunction with the EXPLORE-NFC board and the Raspberry Pi Model B.

This document provides a step by step guide to the installation procedure of the software and the hardware.

Fig 1. EXPLORE-NFC and MIFARE Ultralight card

1.1 What is EXPLORE-NFC?

EXPLORE-NFC is a high performance fully NFC compliant expansion board for the Raspberry Pi. Based on the NXP PN512 solution, EXPLORE-NFC meets compliance with Reader mode, P2P mode and Card emulation standards. The board features an integrated high performance antenna and offers a flexible SPI interface. Software packages supporting all these features are described within this manual.

1.2 What is a Raspberry Pi?

The Raspberry Pi is a credit card sized computer. The initial idea behind it was to develop a small and cheap computer to be used by kids all over the world to learn programming. In the end it became very popular among developers all over the world.

The heart of the Raspberry Pi is a SoC (System on Chip). This contains an ARM11 running at 700 MHz and a graphics processor that is capable of BluRay quality playback,

using H.264 at 40MBits/s. It has a fast 3D core accessed using the supplied OpenGL ES2.0 and OpenVG libraries. In addition, the Model B has 512MB RAM included in its SoC.

To get started quickly, the Rapberry Pi Foundation provides several preconfigured Linux distributions.

For the project described in this guide we are using the “Raspbian wheezy [7.].

For more information about the Raspberry Pi please visit [9.].

1.3 NXP Reader Library and Linux

The projects described in this guide are based on the NXP Reader Library [8.]. The NXP Reader Library is written in C language enabling the customers to create their own software stack for their contactless reader.

To get the NXP Reader Library to work on Linux, some of the relevant parts have been ported to Linux. At the moment, only the SPI protocol is ported for the communication between reader IC and the Raspberry Pi.

The intention of this project is to give an exemplary implementation of the NXP Reader Library on Linux. It should easily be possible to apply/install the Reader Library of this project on other Linux systems.

2. Software installation

2.1 Required items

- Compatible SD card [1.] with a size of at least 4GB.
- Card reader

2.2 Downloading the software

Please download Win32DiskImager and the Raspbian “wheezy” image from the download page of the Raspberry Pi foundation [12.].

2.3 Preparing the SD card

1. Unzip the archive containing the system image.
2. Insert the SD card into a card reader connected to the PC.
3. Start Win32DiskImager.
4. Chose the unzipped *.img file.
5. Ensure that the correct device is chosen.
6. Click onto the Write button.

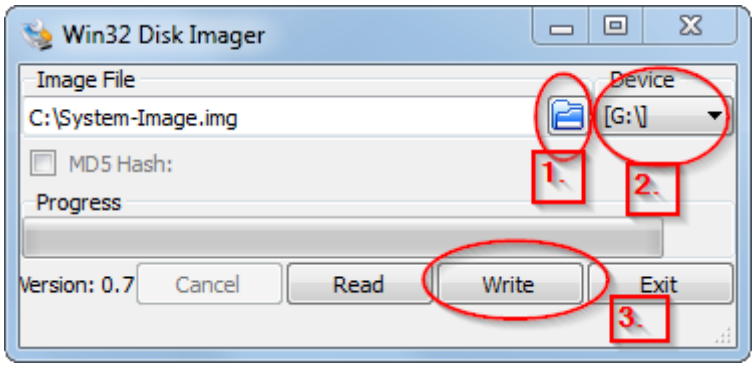

Fig 2. Win32 Disk Imager

7. Copy the zip compressed project onto an USB Stick.

3. Hardware installation

Required items

- Raspberry Pi Model B
- Contactless reader board EXPLORE-NFC
- Compatible SD card [1.]
- Micro USB power supply (5V / 1A) [2.]
- Keyboard
- Mouse
- DVI cable to connect to a Monitor / TV

Please connect the hardware in the following order:

1. Connect the EXPLORE-NFC board.
   Caution: Please take care to connect the extension-board in the right way! See the next picture for more information.
2. Put the SD card into the card reader of the Raspberry Pi.
3. Connect the Raspberry Pi to a Monitor.
4. Connect the Keyboard and the Mouse to the USB ports of the Raspberry Pi.
5. Connect the USB power adapter to the Raspberry Pi and plug it into a socket.

Please wait until the Raspberry Pi is booted and the login screen appears.

Fig 3. EXPLORE-NFC and Raspberry Pi stacked together – top view

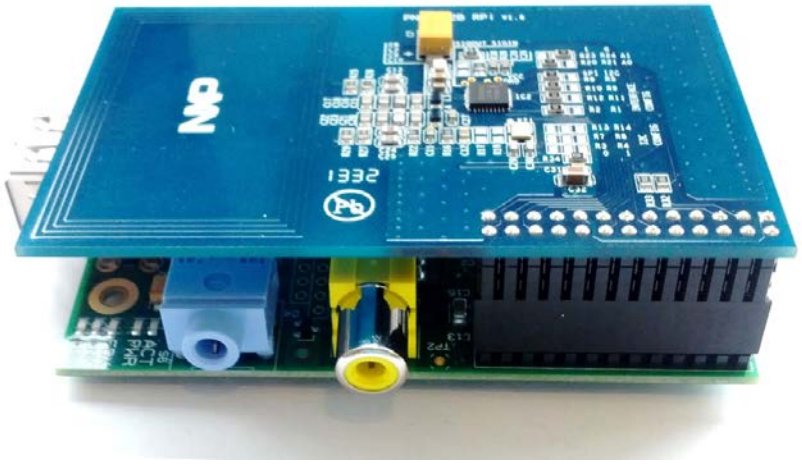

Fig 4. EXPLORE-NFC and Raspberry Pi stacked together – side view

4. Preparation of the demo software for the Raspberry Pi

All example software projects are being delivered within an installer. We provide installer for the platforms Linux x86 and Windows x86. The installer cannot be executed on the Raspberry Pi because of the ARM architecture.

To obtain the example projects, please execute the installer. After the installation routine, the software should be, depending on the operating system, in the following directory:

- **Windows**
  C:\Users\<your username>\NXP Semiconductors\<project name>
- **Linux**
  /home/<your username>/NXP_Semiconductors/<project name>

For example under Windows the source code for the Polling project can be found in the path
C:\Users\<your username>\NXP Semiconductors\EXPLOREPolling-1.0\card_polling.zip

For information about how to use the source code, please continue with the next section in this start guide.

5. Login to the Raspberry Pi and starting the software

This section describes how to compile and run the software for the EXPLORE-NFC demo board. The steps shown here are especially for the Polling project. In principle the description can be transferred to the other projects easily. The only procedural differences will be the naming of the folders.

5.1 Login to the Raspberry Pi

1. Type in username and password, whereas the **username** is **pi** and the **password** is **raspberry.**
2. After the command line appears type in **startx** and hit [Enter]. Wait until the graphical desktop environment has finished loading.

5.2 Activate SPI

If this is the first start, the configuration menu for the Raspberry Pi will appear automatically. Otherwise type the following command to open that menu:
sudo raspi-config .

The option to activate SPI can be found as follows:
Advanced Options → SPI → <Yes>.

Now reboot your Raspberry Pi by typing the following command:
sudo reboot .

5.3 Install additional required software from the package repository

To compile the projects, cmake has to be installed. This can be done by issuing the command

sudo apt-get update && sudo apt-get install cmake

Note: Your Raspberry Pi has to be connected to the internet in order to obtain the required packages.

5.4 Starting the Polling software

1. Copy the zip compressed project onto an USB Stick.
2. Connect the USB Stick with the software project to the Raspberry Pi and unzip the project to the home folder of the pi user.
3. Start the terminal emulator with double clicking on the symbol **LXTerminal** on the desktop.
4. Change into the correct directory (in this example it is ~/card_polling/build) with the command
   cd ~/card_polling/build
5. Configure the software with the command **cmake ../source**
6. Compile the project with the command **make**
7. Start the polling application with the command **./card_polling**

8. The software detects MIFARE cards in an endless loop.
   The **EXPLORE-NFC** board in conjunction with this software is able to detect MIFARE Ultralight, MIFARE Classic, MIFARE Plus and MIFARE DESFire cards.

To stop the software, please press Ctrl + C on the keyboard.

6. How does the Polling software work?

Imagine in a Hotel you got a key card for your room and want to check what card it is. Is it contactless card? And if it is a contactless card, what type of card is it? Or if you got a contactless public transport monthly pass. With the polling software you can get this information immediately.
Another goal is to show the developer how to start a communication with different contactless cards. Based on the software one can easily see the initialization flow that is needed to be able writing or reading contents to or from cards.

So the basic idea behind the polling software is to identify the exact type of a contactless card.

6.1 Functional principle

The main software is divided into three parts.
First it checks for command line arguments. After that is done, the software initializes the hardware specific parts according to the chosen hardware.
The **second** step is to initialize some more hardware specific but common parts like the SPI interface. If everything went fine, the polling procedure can start.
Third, the software begins to check if any supported NFC card is in the field. It begins with the MIFARE cards, than the JIS X 6319-4 compatible cards and it checks if there is a ISO/IEC 14443B card in the field.

The initialization of the required protocols before the detection can be started, and the detection itself, is done in separate functions.

If any NFC card has been detected, the software stops looking for other cards and prints out the type of the found card. For example: if the software detects an MIFARE card, it does not look for JIS X 6319-4 compatible cards and ISO/IEC 14443B cards.

6.2 Card detection

The project can be split into four different card and tag detections. It can detect MIFARE cards, FeliCa cards and ISO/IEC 14443B cards.

The function “DetectMifare()” is responsible to detect all MIFARE cards and determine their type [15.].

The function “DetectFelica()” is responsible to detect all JIS X 6319-4 compatible cards.

The function “DetectTypB()” is responsible to detect all ISO/IEC 14443B cards.

6.3 Detecting Payment cards

The Polling project is also able to detect cards or tags and return their type. Additionally, when a Payment card is detected, this project also tries to get the Payment System Environment (PSE) and the Application Identifier (AID) of the card.

Three functions are used for the additional part. The function “PaymentCard” tries to get the PSE and AID while the functions “Compare” and “Card_Scheme” help this function to check whether the PSE and AID are valid or not.

The function “PaymentCard” sends a request to get access to the PSE. The PSE is a file either with the name 1PAY.SYS.DDF01 or 2PAY.SYS.DDF01. With two possible names there are two possible commands that can be sent. The arrays containing the request

commands are named “AppSelection1” and “AppSelection2”. Only one of those commands need to be sent to the card If both tries to select the PSE file are answered with an error, we can assume that we have no PSE crad in the field.

When the return value is positive, the PSE name will be taken from the return answer and the function “Compare” checks whether it is valid or not.

If the PSE is valid, the return answer will be analyzed until the AID is found. Then the function “Card_Scheme” will be used to determine the card type. The type of the card will be returned afterwards.

The function “Compare” checks the PSE of the card with the given PSE values 1PAY.SYS.DDF01 and 2PAY.SYS.DDF01. If the PSE values do not match, there is no PSE on the card and the function returns an according value.

The function “Card_Scheme” retrieves the AID of the card and checks it with a list of known AIDs and then returns a number.

These AIDs and their matching card types can be obtained from Wikipedia [16.].

7. Card Emulation project

Suppose you want to organize an outdoor game similar to Paper Chase and you want to give it a digital touch. In addition you also want to give some audience the opportunity to follow the game live.

Then Card Emulation is the key technology for that purpose. The idea behind using it for Paper Chase is that the organizer may place a Raspberry Pi with the EXPLORE-NFC board and a mobile internet connection next to every riddle. A player that solves the riddle can than send the solution with his own NFC enabled mobile phone contactless to the Raspberry Pi. The application on the Raspberry Pi then can decide how to process or where to publish the solution. That makes it possible for people outside following the progress of the Paper Chase game. In the example this guide refers to, the messages are being posted on Facebook.

Another use case could be a feedback station in a restaurant. With that approach a restaurant manager could set up the software on the Raspberry Pi accordingly and give the customer a way to give feedback with his own mobile phone very easily.

The main purpose for Card Emulation is to exchange a small amount of data between two devices contactless over NFC, e.g. the Raspberry Pi with the EXPLORE-NFC board and an NFC enabled mobile phone. In Card Emulation mode you can exchange all kind of data that can also be written onto an NFC card.

7.1 How does it work?

In total there are three applications involved in this project.

- An application on an Android mobile phone. This application provides the functionality to read, compose and send messages to the Raspberry Pi over NFC.
- An application on the Raspberry Pi that emulates a contactless card for the data exchange over NFC.
- A second application on the Raspberry Pi written in Java that receives the message from the Card Emulation application on the Raspberry Pi and posts it over the Internet on Facebook.

The application, that takes the message from a smart phone to be posted to Facebook, emulates an ISO/IEC 14443A Type 2 Tag. For detailed specification please refer to the web page of the NFC Forum [17.]. General description of the Card Emulation project

The PN512 supports 4 different operating modes:

- Reader/Writer mode supporting ISO/IEC 14443A/MIFARE and FeliCa compliant scheme
- Reader/Writer mode supporting ISO/IEC 14443B
- Card Operation mode supporting ISO/IEC 14443A/MIFARE and FeliCa compliant scheme
- NFCIP-1 mode

The card operation mode is passive mode, in which the PN512 does not generate an RF field but acts as a card that modulates the field for communication with the reader.

A specification to store data for any kind of service and application is specified in the NFC Forum and it is called NFC Data Exchange Format. Storing NDEF formatted data inside contactless card products as mapping models as well as the management of NFC forum device as a specific platform such as a NFC Forum Type 4 Tag are defined in [18.]. The following project shows an exemplary implementation of a Tag 2 Type Card on the PN512.

7.2 Configuration of the example project

In order to change some of the possible options before compiling the project, the file "src\NxpRdLib_PublicRelease\intfs\phCardEmu_Options.h"
should be edited. This file contains toggles to change some default values like memory sizes or pin numbers in which the PN512 chip is connected to the Microcontroller. Further options are also indicated in the file.

Note: To get the provided application running, you only need to make changes in the file settings.properties. This file provides the access credentials for the Java application to Facebook.

7.3 Building and starting the project

Before you can work with the Card Emulation project you need to make sure that the following additional packages are installed on the Raspberry Pi:

- cmake
- oracle-java7-jdk

You can install these packages with the command **sudo apt-get update && sudo apt-get install cmake oracle-java7-jdk**.

Once these packages are installed, you can build the project as follows:

1. Change into the directory build
2. Type the following command: **cmake ../source**
3. Compile the source with the command **make**
4. Edit the file settings.properties and enter your credentials for Facebook.
5. Run the program with the command **./cardEmulation-fb**

7.4 Changing the predefined NDEF message

The predefined NDEF message can be changed in the following files:

1. Open the file
   ..\src\NxpRdLib_PublicRelease\comps\phceT2TCmdHdl\src\Sw\phceT2T_Const.c
2. Change the NDEF message defined in `const uint8_t T2T_DATA_DEF[].`

Which one of the messages is used by the software is defined with the line
`#define T2T_NDEFFILE_PREDEF  4.`

7.5 Source code of the Java and Android applications

The sources of all applications involved in that example are provided with these examples. Both applications, the Android application and the Java application, are provided as Eclipse projects.

7.6 Program flow

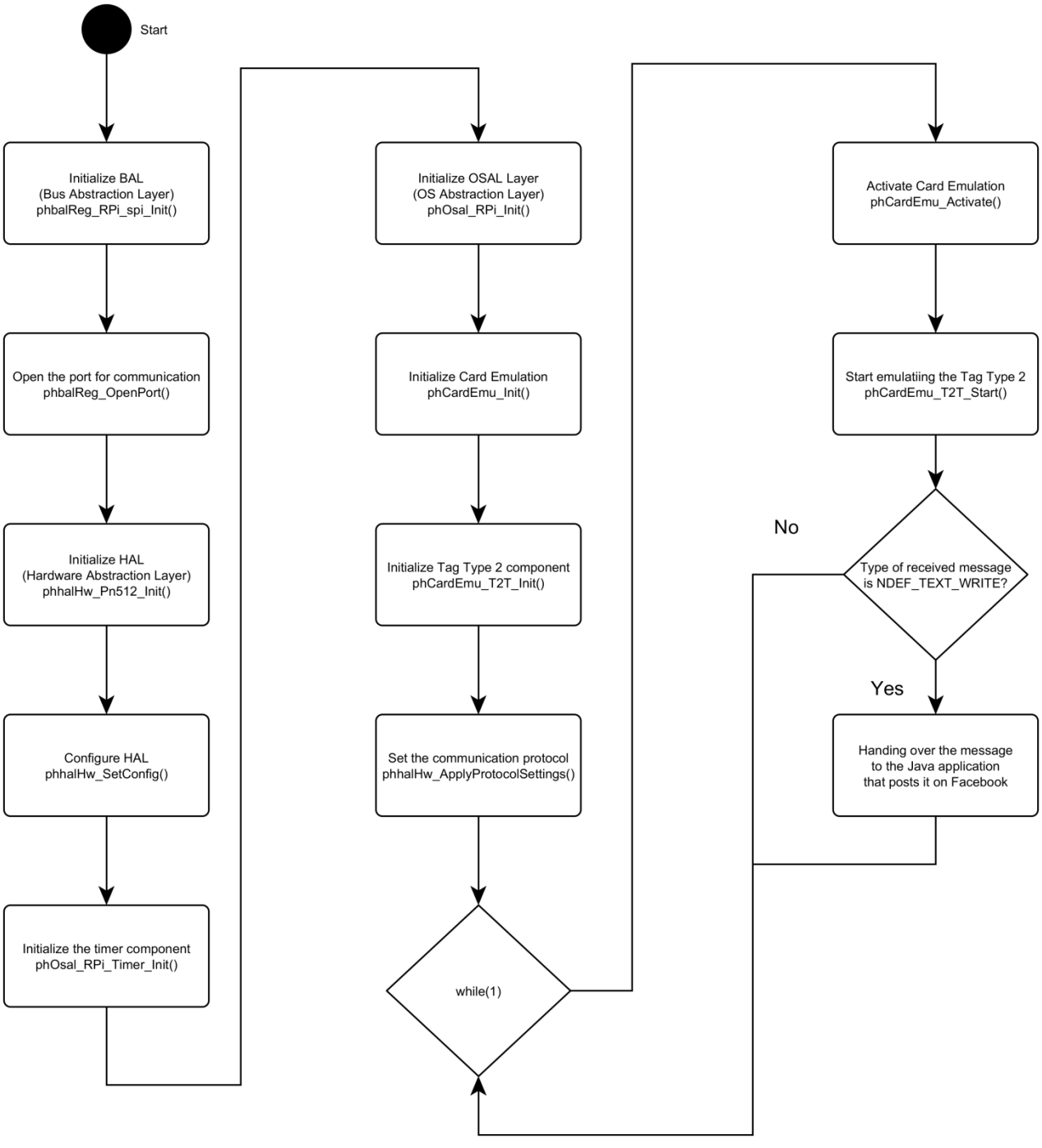

Fig 5. Program flow of the Facebook Card Emulation example

The first blocks describe the initialization of the necessary layers and components independent of the card emulation. After the layers have been initialized, the card emulation can be initialized too. After applying the protocol settings, the PN512 waits for a successful activation of the card side. This activation can be done for example by an NFC enabled mobile device.

Now the reader on the opposite can read the NDEF message that is presented from the PN512. If this message gets modified, for example by an Android mobile, the card emulation application passes that message to the Java application that posts it on Facebook.

As soon as the Java application is done, the NFC application again waits for the next message.

7.7 Android Application

7.7.1 The Application

Basically, the application is meant for reading or writing of NDEF messages from either an emulated card or an NFC Smart Card. On the main screen you can choose between the two screens “Read” or “Write”. With the “Read” screen you can read NDEF messages that have already been stored on the emulated card whereas the “Write” screen enables you to browse the message store, create, delete and modify messages.

7.7.2 Installing the Application

Connect your phone with the computer (Micro USB). Now you can see your phone in the Windows Explorer. Open the internal store.

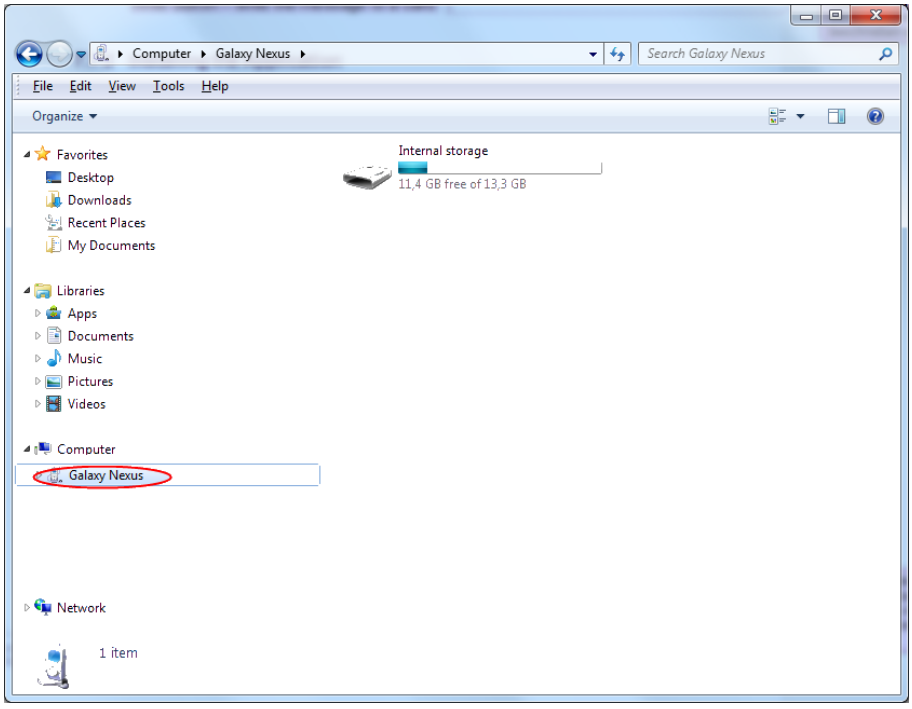

Fig 6. Android phone accessible in Windows Explorer

Choose a folder where you want to save the file and copy the application file (FacebookMsg.apk) there.

Take the phone and open the data manager application. Switch to the selected folder and tap the Facebook.apk file.

Tap the “Settings” button to accept unknown sources. Unknown source means that the application has not been obtained from the Google Play Store.

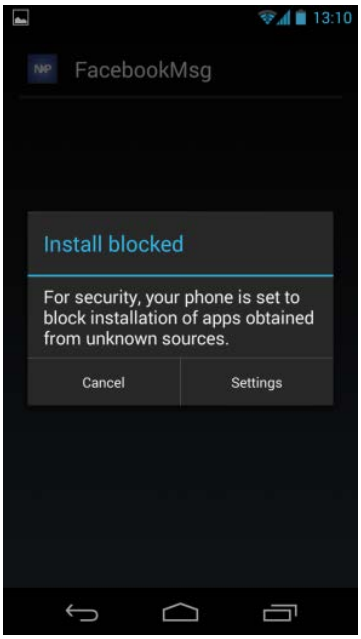

Fig 7. Install blocked message warning

To accept the unknown source tick the checkbox on “Unknown sources”

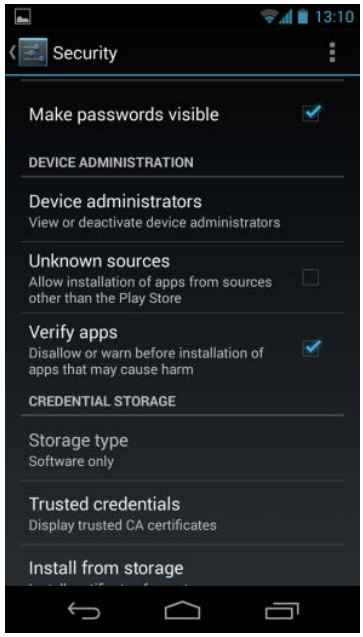

Fig 8. Accept unknown sources

Change to the file manager again and tap the “FacebookMsg.apk”. Now tap the “Install” button.

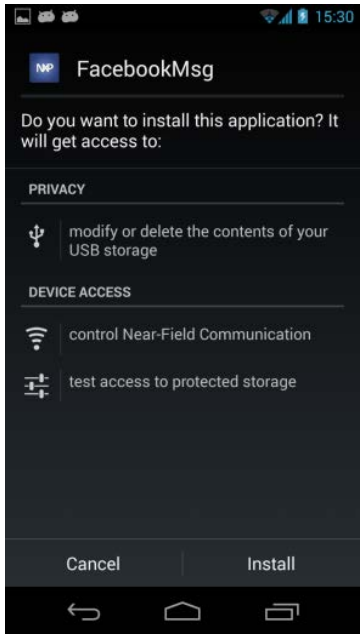

Fig 9. Install the Application

Installation finished. Tap “Open” to use the application.

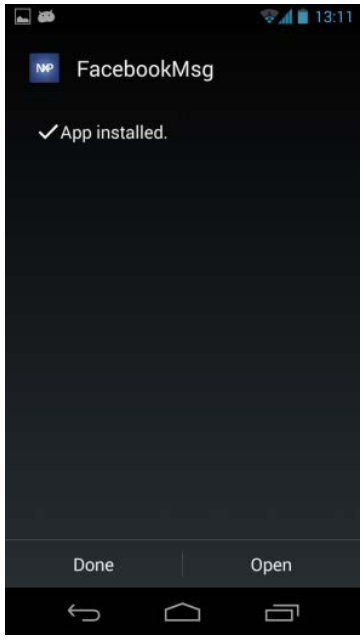

Fig 10. Finish

7.7.3 First post

Choose the “Write“ screen.

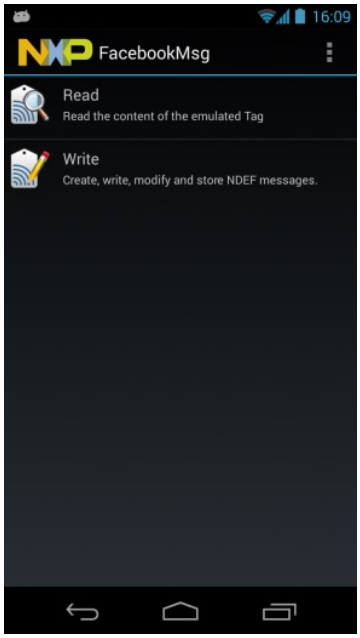

Fig 11. Startscreen

Tap the „Create“ button for creating a new message

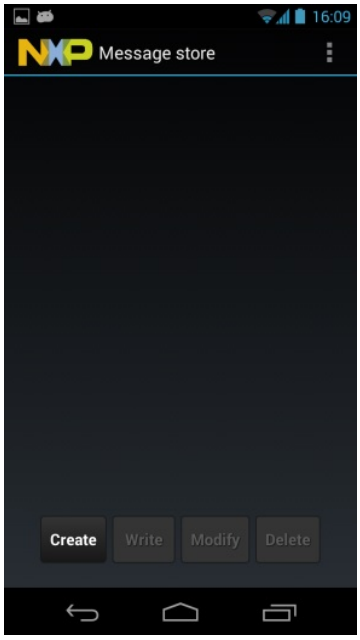

Fig 12. Message store

Type your message and tap the “Store” button

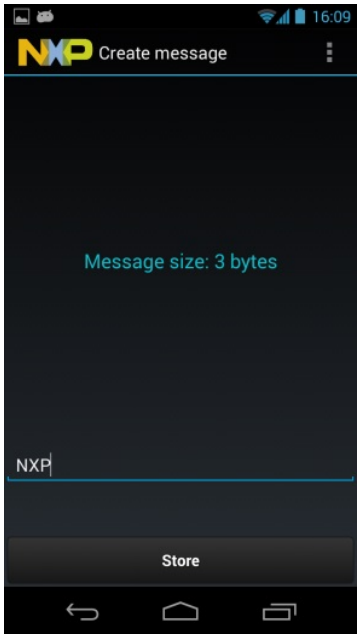

Fig 13. Create a message

Select your message in the message store and tap the “Write” button.

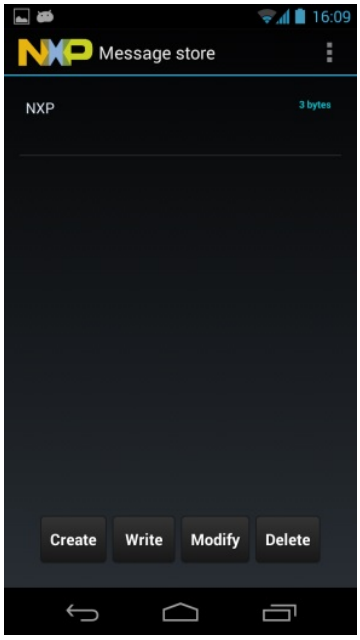

Fig 14. Message store

Touch the antenna of the NFC extension board with your device

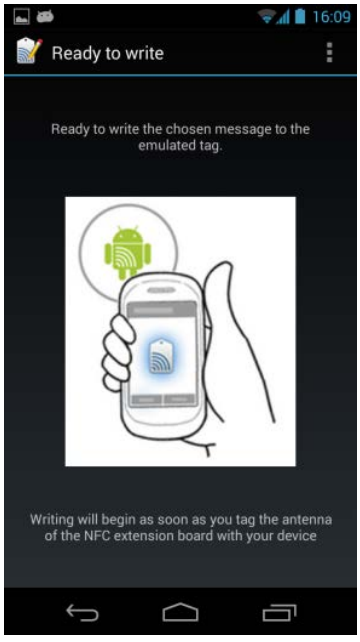

Fig 15. Write screen

Now the message gets sent to the NFC application that has been executed on the Raspberry Pi previously.

8. P2P SNEP client project

Imagine you receive an image via e-mail and want to upload it to your mobile phone. If you have no cable at hand, you need to fiddle about with Bluetooth or WiFi.

That is the point where NFC – P2P makes the game. With the P2P SNEP client you can easily transfer the image by just touching the NFC antenna with your mobile phone. No need to set up the connection before. No fiddling with cables or configuration. Because the NFC reach range is limited to only a few centimeters, the NFC app assumes that it is the user's intention to establish a data connection and transfer the data.

The main purpose of P2P is to exchange a large amount of data between two devices over NFC in a contactless way, e.g. the Raspberry Pi with the EXPLORE-NFC board and an NFC enabled mobile phone. In P2P mode you can exchange all kinds of data. In the example this guide refers to, we are using P2P with the PUT command to send an image file from the Raspberry Pi to an NFC enabled mobile phone. Before sending the image it is being encapsulated into an NDEF message of the type image/jpeg.

8.1 How does the SNEP client software work

It sends an NDEF message encapsulated in a SNEP message to the mobile device, which must be capable of performing NFC and the peer to peer standard. Compliance with the LLCP and ISO18092 protocols are ensured by the P2P Library. Execution of the SNEP client software can be summarized in the following steps:

1. Hardware initialization
2. Detecting the RF field for an NFC peer of the tag type F.

   The software checks the RF field whether there is tag type F capable of performing the P2P communication
3. Once such device is found, the LLC link is activated in compliance with the procedure defined by the NFC forum [11.].
4. LLCP socket creation and establishing connection with other peer – SNEP server.
5. Transmission of a given image file to the SNEP server:

   The SNEP client sends an initial fragment 128 bytes long. Then it waits for a response from the server. Because in SNEP header it is declared longer SNEP message than one fragment, the server should response with the Continue response. The SNEP client can go on with sending the rest of the SNEP message. As soon as the entire SNEP message has been transmitted, the SNEP client shall receive the SNEP Success response from the mobile device and the transmitted picture should be displayed on the mobile's screen.

   Obtaining of the

8.2 Choosing the NDEF message

By default the software sends an image of the NXP logo as NDEF message. There are more NDEF messages prepared in dedicated header files (see Table 1). Only a single header can be compiled with the SNEP client application. To choose another content of the NDEF message for transmission just do following modifications in the **Error! Unknown document property name.***/source/ndef_message.c*.

1. Open the *ndef_message.c* source file in a text editor.
2. Uncomment the line with a header file to be transmitted. Let all the other lines commented.
3. In Table 1 in the same row as the chosen file look up two identifiers corresponding to the chosen file.
4. In the array `nmess[]` uncomment the line with the couple of identifiers corresponding to the type of the chosen file. Comment all the other lines.
5. Save the changes and close the text editor.

Table 1. Table of files prepared for NDEF message transmission

Identifiers from the last two columns are necessary for choosing the right line from `n_mess[]`.

Content	Header Name	NDEF message identifier	File type identifier
PNG image	*c_tablepng.h*	`NDEF_TYPE_IMAGE`	`NDEF_IMAGE_PNG`
QR code of NXP	*c_tableQR.h*	`NDEF_TYPE_IMAGE`	`NDEF_IMAGE_PNG`
Image of NXP logo	*c_tablenxp.h*	`NDEF_TYPE_IMAGE`	`NDEF_IMAGE_JPEG`
Long text message	*c_tabletxt.t*	`'T'`	`LANG_EN`

8.3 Folder structure of the project

In the project directory **Error! Unknown document property name.** there are two directories

/**source** containing

- all the required source files for performing the SNEP client
- including the NXP Reader Library P2P
- *CMakeLists.txt* script.

/**build** containing

- Before compilation, this folder is empty. Afterwards object files and the executable SNEP client file shall be here.

8.4 Building and starting the project

Before you can work with the P2P project you need to make sure that the following additional package is installed on the Raspberry Pi:

- cmake

You can install this package with the command **sudo apt-get update && sudo apt-get install cmake**.

Once this package is installed, you can build the project as follows:

6. Change into the directory build

7. Type the following command: **cmake ../source**
8. Compile the source with the command **make**
9. Run the program with the command **./Snep_client**

8.5 Timers

In the NXP Reader Library P2P there is the OSAL module – *NxpRdLib_PublicRelease/comps/phOsal* encapsulating memory allocation and time related functions. Time related functions are more precisely described in [10.]. Basically there are two groups:

Timer handling functions for general usage of the timers.

Waiting Delay – the current task stops execution for a given amount of time.

8.5.1 Importance of timers

The SNEP client communicates on the Logical Link Control Protocol (LLCP). In the LLCP there is Link Timeout defined which ensures periodical exchange of the Symmetry token to determine and avoid a termination of the logical link.[11.] Therefore at least one timer is utilized by the SNEP client application. Obviously the timers can be used by the user for any purpose.

8.5.2 Selection of the timers

Depending on the used hardware platform the correct part of the OSAL module need to be compiled.

Since this application note deals with the Raspberry Pi board, this option needs to be configured.

Raspberry Pi board

Uncomment in *ph_NxpBuild.h:* `NXPBUILD__PH_OSAL_R_Pi`.

Comment in *ph_NxpBuild.h:* `NXPBUILD__PH_OSAL_12XX`

Implementation: raspbian wheezy OS software timer and signal functions from *time.h* and *signal.h* respectively. See implementation of the software timers and handling the signals in the source file *comps/phOsal/src/RaspberryPi/phOsal_R_Pi.c*. See the headers *time.h* and *signal.h* are included.

This option is mentioned only for comparison, if interested.

9. Supplementary notes

9.1 Turning off the Raspberry Pi

The Raspberry pi has no power switch. To turn it off, please type **sudo halt** into the command prompt. After the screen turns black, you may unplug the power supply.

9.2 Resizing the file system

In case one uses an SD card bigger than 2GB it might be reasonable to resize the root file system to use the whole space of the SD card. To do so, please follow these steps:

1. Start the Raspberry Pi and log in.
2. Type **sudo raspi-config** into the command prompt.
3. In the configuration dialog that opens, choose the entry “expand_rootfs” and press [Enter].

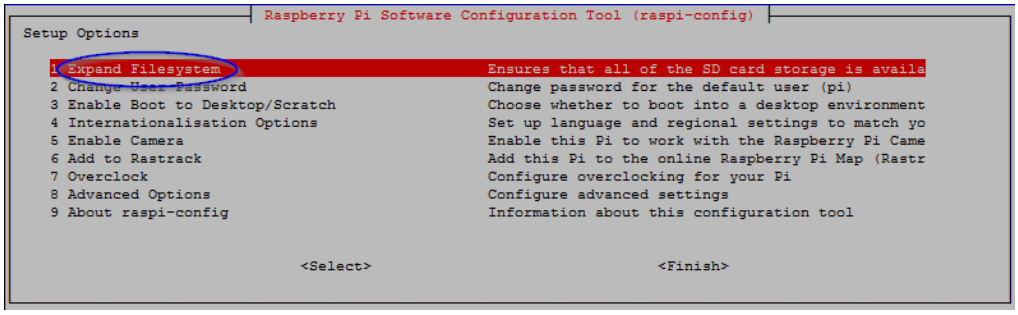

Fig 16. Expand rootfs in Raspi-config

4. After the operation has finished, choose <Finish> by pressing the [TAB] key twice and press [Enter].
5. Answer the question about the reboot with <Yes>.

10. References

[1.] **List of verified SD cards:**
http://www.nxp.com/redirect/elinux.org/RPi_SD_cards

[2.] **List of verified USB power adapters:**
http://www.nxp.com/redirect/elinux.org/RPi_VerifiedPeripherals_Power_adapters

[3.] **Software for the Raspberry Pi:**
https://nxp.box.com/RaspberryPi

[4.] **Supported wireless network adapters:**
http://www.nxp.com/redirect/elinux.org/RPi_VerifiedPeripherals_USB_Wi-Fi_Adapters

[5.] **Some general information about the Raspberry Pi**
http://www.nxp.com/redirect/raspberrypi.org/faqs

[6.] **Downloads for the Raspberry Pi**
http://www.nxp.com/redirect/raspberrypi.org/downloads

[7.] **Raspbian Linux**
http://www.nxp.com/redirect/raspbian.org

[8.] **Direct link to the NXP Reader Library**
http://www.nxp.com/documents/software/200310.zip

[9.] **FAQ for the Raspberry Pi**
http://www.nxp.com/redirect/raspberrypi.org/faqs

[10.] **User Manual P2P Library CLRC663, PN512**
http://www.nxp.com/documents/user_manual/UM10721.pdf

[11.] **Technical Specification Logical Link Control Protocol, NFCForum-TS-LLCP_1.1**
http://www.nxp.com/redirect/nfc-forum.org/specs/spec_license

[12.] **Download page of the Raspberry Pi foundation**
http://www.nxp.com/redirect/raspberrypi.org/downloads

[13.] **EMV – The table of card types and their matching AIDs are available on**
http://www.nxp.com/redirect/en.wikipedia.org/wiki/EMV

[14.] **AN10833 MIFARE Type Identification Procedure – Description of how to detect the type of a MIFARE card**
http://www.nxp.com/documents/application_note/AN10833.pdf

[15.] **MIFARE Products**
http://www.nxp.com/redirect/mifare.net/en/products/mifare-smartcard-ic-s

[16.] **List of AIDs at Payment Cards**
http://www.nxp.com/redirect/en.wikipedia.org/wiki/EMV_Application_selection

[17.] **NFC Forum Tag Type Technical Specifications**
http://www.nxp.com/redirect/nfc-forum.org/specs/spec_list/_tagtypes

[18.] **TYPE 4 TAG: NFC Forum, Type 4 Tag Operation Specification, Version 1.0, March 13, 2007**
www.nxp.com/redirect/nfc-forum.org/specs

11. Legal information

11.1 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.

11.2 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.

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 accepts 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.

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.

11.3 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.

11.4 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.

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

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

DESFire — is a trademark of NXP B.V.

12. List of figures

13. Contents

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

Date of release: 17 December 2013
279710
Document identifier: AN11480