



FTF 2016
TECHNOLOGY FORUM

USE NFC PAIRING TO CONNECT MULTIPLE BLUETOOTH AND WIFI DEVICES WITH A SIMPLE TAP

DARRIN PATEK
APPLICATIONS ENGINEER
SESSION FTF-HMB-N1900
MAY 18, 2016

PUBLIC USE



AGENDA

- Introduction to NFC, background, applications and key points
- The NFC devices used in today's lab and the basics of how they communicate
- Brief introduction to the Linux, Android and RTOS NFC stacks
- Overview of the PN7120
- High level overview of the i.MX 6SoloX and lab hardware
- Today's Lab
- NFC is cool where do I learn more?

NFC

BACKGROUND, APPLICATIONS AND KEY POINTS

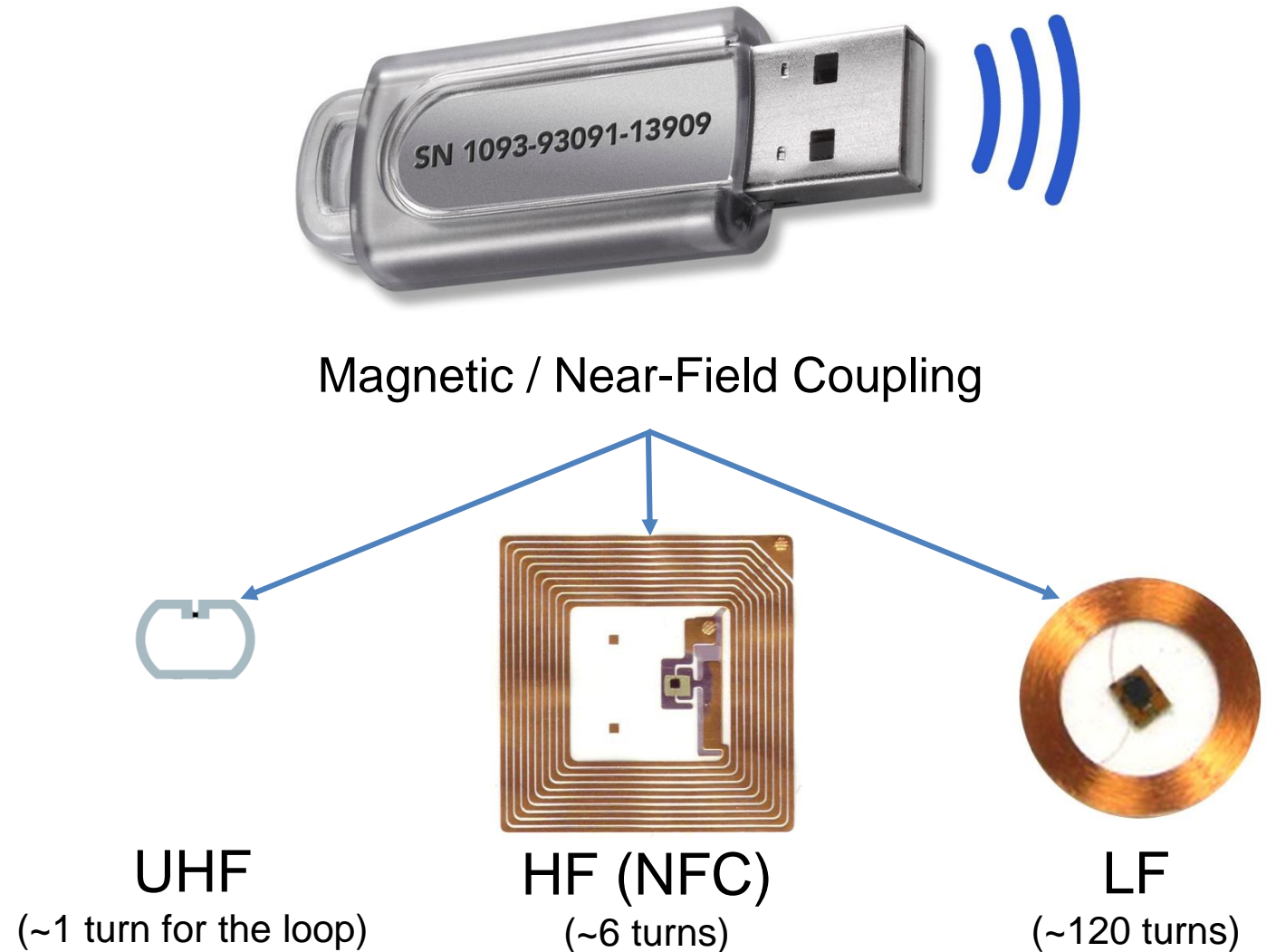
What is NFC and Why Was It Created?

Near Field Communication is a short-range wireless connectivity technology *standard*, designed for *intuitive* and *simple* communication between *two* electronic devices.



Passive RFID Tags

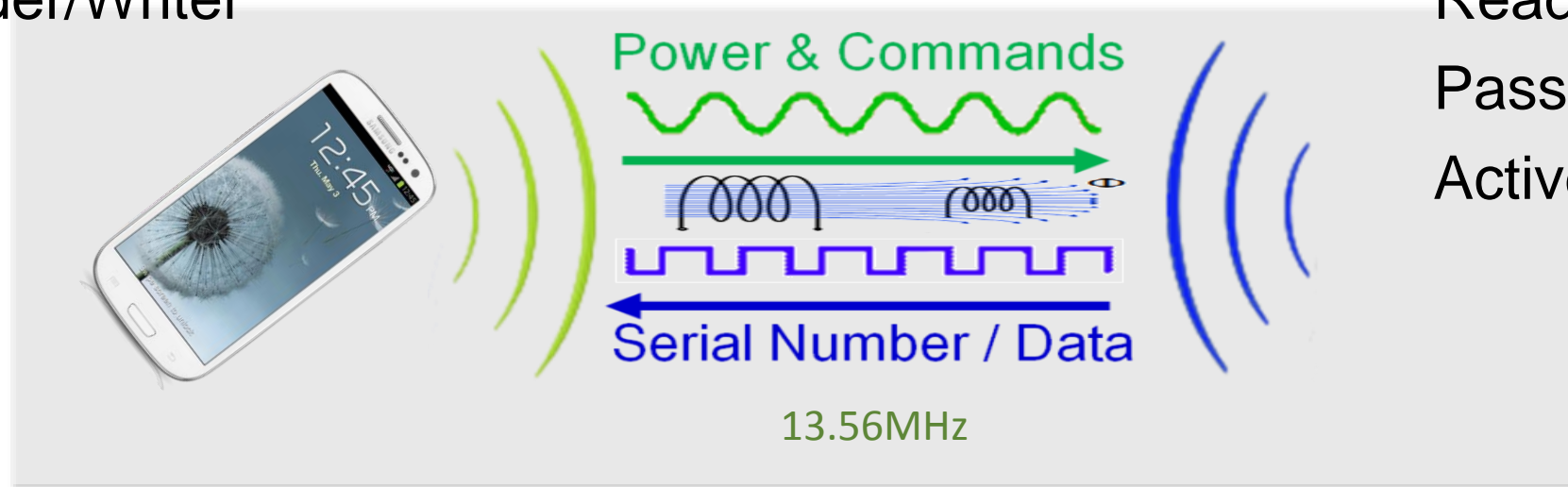
- Like a wireless (read/write) memory stick with a unique serial number to identify items
- Three primary frequencies:
 - Low Frequency (LF-125kHz)
 - High Frequency (HF-13.56 MHz)
 - Ultra High Frequency (UHF-840 - 960 MHz)
- Frequencies of discussion:
 - High Frequency (HF/NFC)



NFC | Principal of Operation

- Same fundamental principle as RFID, contactless smart cards or access control badges
- Reader (e.g. mobile device) provides power, initiates RF communications & captures data from the tag (or programs data into the tag)

Reader/Writer



Reader/Writer

Passive Tag

Active Tag

RFID, Proximity & NFC

We hear often about them, but what is covered with these names



RFID: Radio Frequency Identification

- Generic term for contactless technology.
- Always used for applications related to **tagging of goods and items**
- Range from few cm to several m – automatic detection of a Unique Identifier (few bytes)
- Based on various technologies : LF (120-150 KHz), HF (13.56 MHz), UHF (433 to 900 MHz)
- Standardized in ISO18000

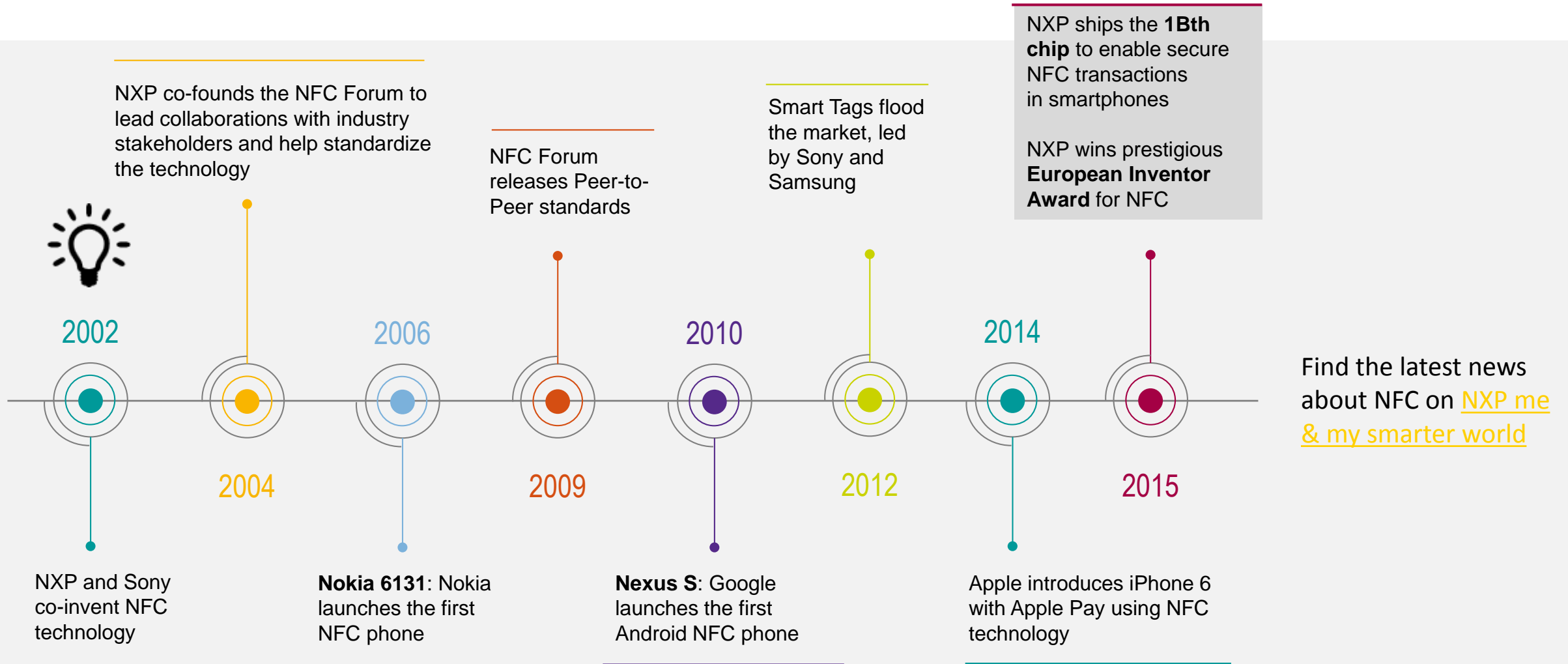
Contactless **Proximity** technology

- Subset of RFID, limited to 1 frequency range : 13.56 MHz. Associated to people, active action required (put card in front of reader)
- Use in **Access Control, Passport, Payment, Transport**
- More memory, more security
- short range (several cm)
- Standardized in ISO14443

NFC: Near Field Communication

- NFC builds on Proximity technology by **extending the technology to Peer-to-Peer** and Card Emulation
- Short range
- Standardized at the NFC Forum

The NFC Evolution

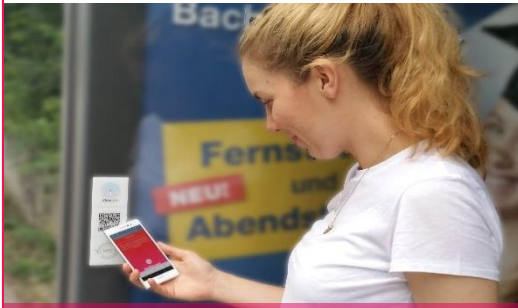


NFC – Three Application Families



Read/Write Mode

- Interacts with an NFC-enabled device
- Reads data in from device or writes data out



Get information or initiate an action



Peer-to-Peer Mode

- Establishes two-way communication between NFC-enabled devices
- Each device serves as an endpoint



Passive and active communication, e.g. pairing



Card Emulation Mode

- System behaves as contactless smartcard*
- Makes NFC-enabled systems compatible with contactless cards



Ticketing, payments, access, control, transit...

* ISO/IEC 14443-compliant smartcard

NFC Applications in Mobile Phones

Payment
Mobile phone = POS



Service Discovery
Take info from poster:
Mobile phone = ticket counter



Transactions
Access Control:
Mobile phone = key



Transactions
Access to
public transport:
Mobile phone =
transport card



Transactions
Micro-payments:
Mobile phone =
debit card



Connectivity
Exchange
information
Mobile phone =
electronic business
card

NFC Applications in Computing and Consumer Electronics

Transactions
Secure Payment



Connectivity
Quick and secure
WLAN/Bluetooth
set-up



Connectivity
Enable quick and easy
long range data transfer

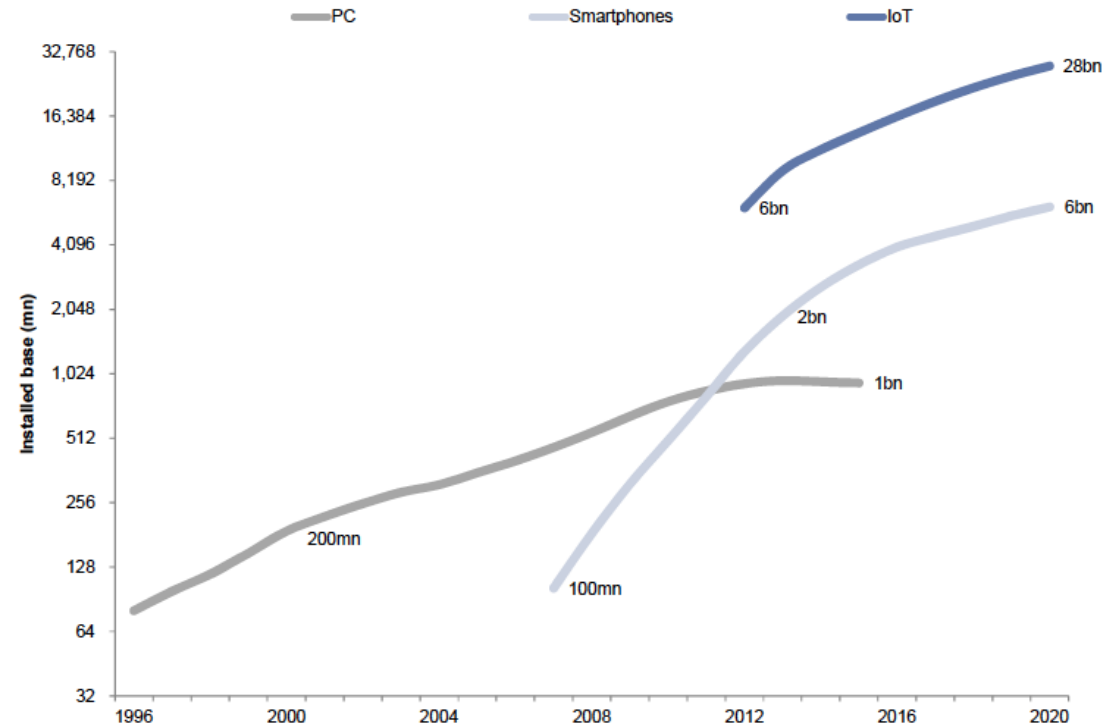


NFC and the Adoption in The Internet of Things (IoT)



Exhibit 4: IoT emerging as the next mega-trend
Internet subscribers over time

Note: y-axis is on a logarithmic scale



Source: IDC, Ericsson, Goldman Sachs Global Investment Research.

PN7120 is the Ideal Solution for Rapidly Integrating NFC Technology In Any Application

- Pairing
 - Bluetooth, Wi-Fi, ZigBee
 - Home automation commissioning
- Personalization
 - Personalize your device, Parental control
- Payment
 - Access VoD content from any provided.
- Logical access control
 - Grant access only to authorized individuals
- User Interface
 - Configure settings with smartphone
 - Data transfer, product registration
- Maintenance
 - Troubleshooting cloud assistance
 - Firmware update
- Authentication
 - Originality check for ink cartridges, batteries, replacements



Set top box/Smart TV



IoT home gateway



Printer



Appliances



NFC DEVICES

USED IN TODAY'S LAB AND THE BASICS OF HOW THEY COMMUNICATE



NFC at a Glance

- Contactless proximity technology
- Operating frequency: 13.56 MHz
- Operating range: 10 cm
- Maximum speed: 848 Kbps (106 Kbps)
- Standardized in ISO/IEC, ECMA and ETSI.
- Compatible with existing ISO/IEC 14443 and FeliCa contactless cards & reader infrastructure
- (Read/Write), Card Emulation and Peer-to-Peer modes possible in one device
- Quick, seamless pairing with Bluetooth, Wi-Fi
- NFC Forum as a key standardization & interoperability group

Read/Write mode: Behaves as a contactless reader



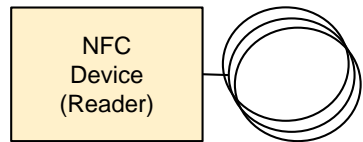
Passive Tag: Behaves as a wireless EEPROM

Active Tag: Behaves as a wireless EEPROM but with the addition of a serial interface.



NFC Communication Modes **Reader/Tag Passive**

Communication Mode



1. Power

The RF field oscillates at 13,56MHz.
The card is powered through the electromagnetic coupling



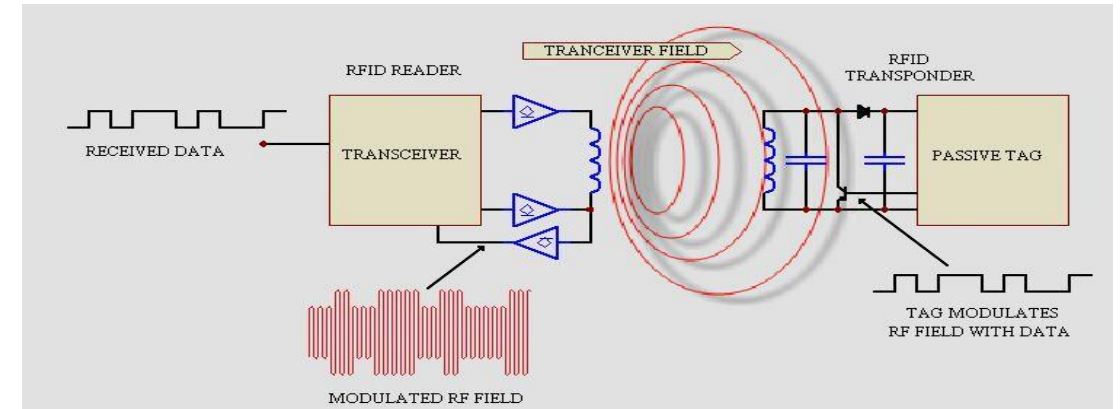
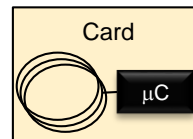
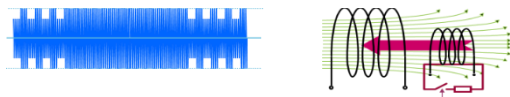
2. The Reader sends commands

The Reader modulates its RF field to send commands

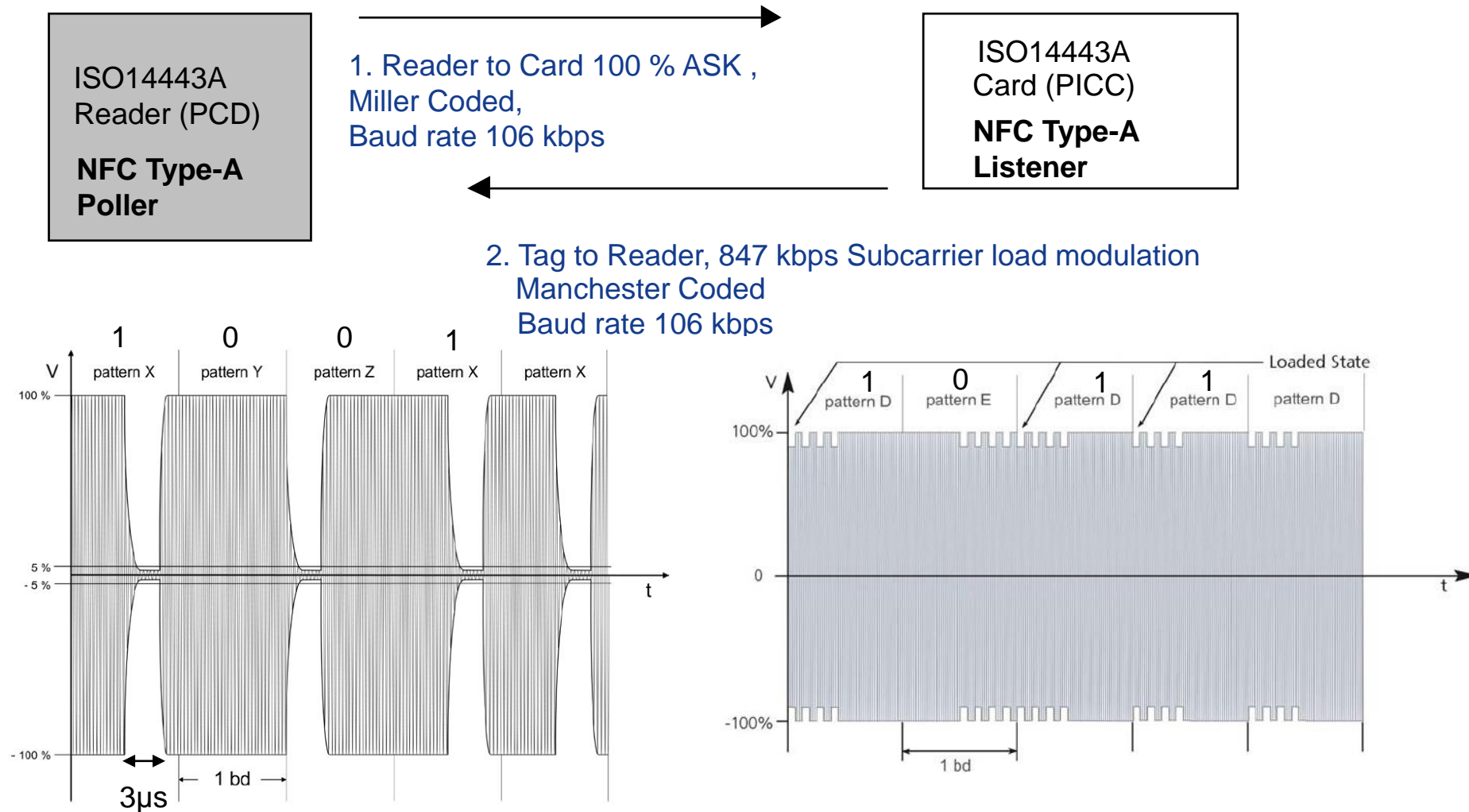


3. Answering to the Reader

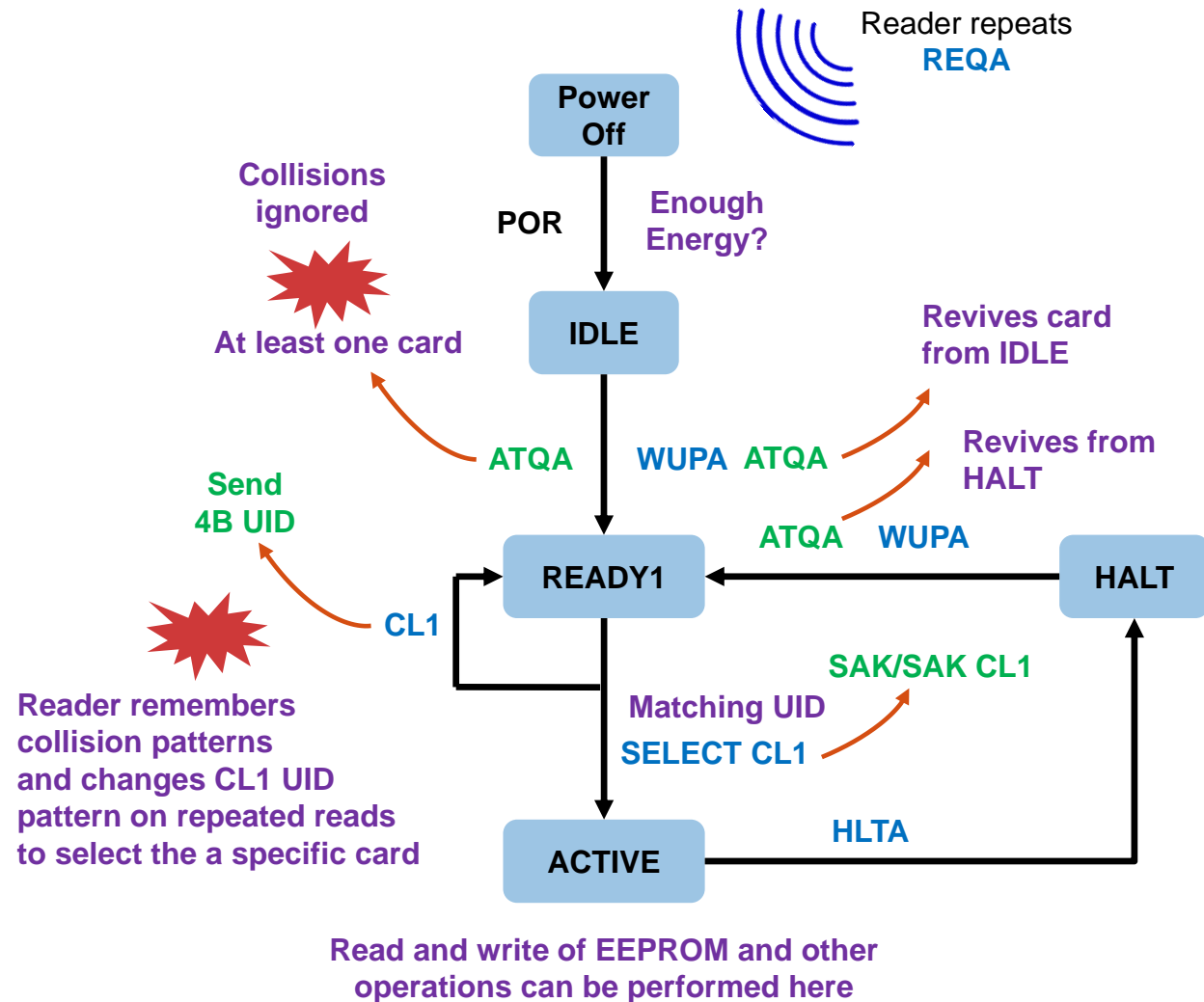
By modifying its consumption, the chip modifies the RF field, which is detected by the Reader (**Load Modulation**)



What Does the Physical Layer Look Like and Why Was It Chosen?



The Type-A Tag Simplified State Diagram Shown with Multiple Tags



- Responses are Bit Synchronous
- Reader commands in Blue
- Tag responses in Green
- Tag state transitions in Black
- Information in Purple

Contactless Smart Card Standards Overview

Physical Layer	Type-A ISO14443A	Type-B ISO14443B	ICODE and ISO15693	Felica (not an ISO standard)
PCD to PICC	100% ASK Miller modified coding	10% ASK NRZ Coding	10% or 100% ASK Manchester Coding	8 – 30% ASK Manchester Coding
PICC to PCD	Subcarrier fc/16 OOK Manchester	Subcarrier fc/16 BPSK NRZ-L	Single or Double Subcarrier fc/32 or fc/28 Manchester	>12% ASK load modulation Manchester Coding

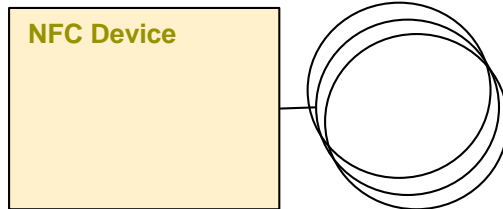
PCD - Proximity Coupling Device (i.e. reader)
PICC - Proximity Inductive Coupling Card (i.e. tag)

<http://www.nearfieldcommunication.org/nfc-signaling.html>

<http://www.nearfieldcommunication.org/tag-types.html>

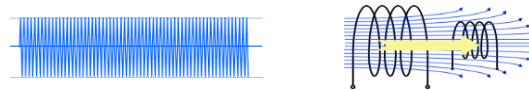
Peer-to-Peer **Passive** Communication Modes

Initiator



1. The Initiator generates the RF field

This field is used to exchange the data. Both Initiator and Target are powered internally



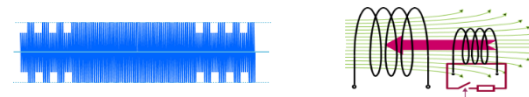
2. Sending commands

The Initiator modulates the RF field to send commands



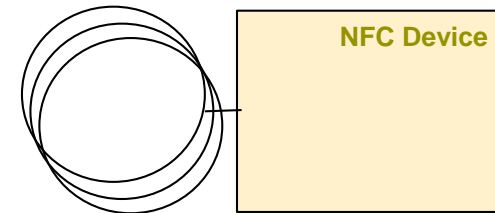
3. Target answers

Similar to the card case , the Target will use a “backward Modulation” to transmit its answer.

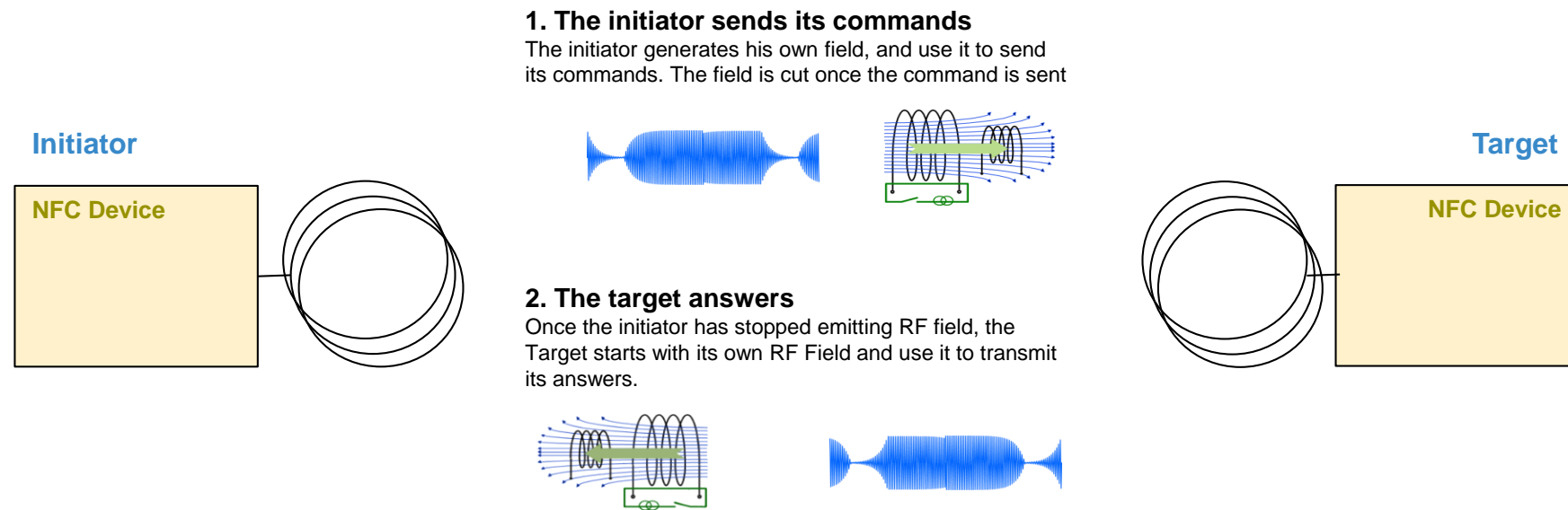


Same concept as for the Read/Write mode

Target

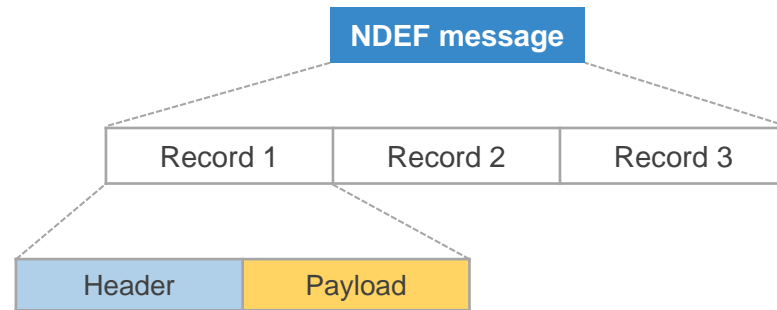


Peer-to-Peer **Active** Communication Modes



Data Format for Communications

NDEF – building an interoperable data exchange



Payload example

- NFC Forum Well-Known Type [NFC RTD]
 - 0x01 : http://www.
 - 0x02 : https://www.
 - 0x05 : tel:
 - 0x06 : mailto:
- Internet media-type (MIME) defined in RFC 2046
 - Image/jpeg, text/html

☞ NDEF messages are the basis of exchange between NFC devices and tags

- NDEF structures the exchange of data during P2P data exchange, in 1 or several records
- NDEF enables an abstraction of the tag memory organization, which is different for each Tag Type.
- Record Type Definition allows the invocation of the appropriate application to handle the NDEF payload contents

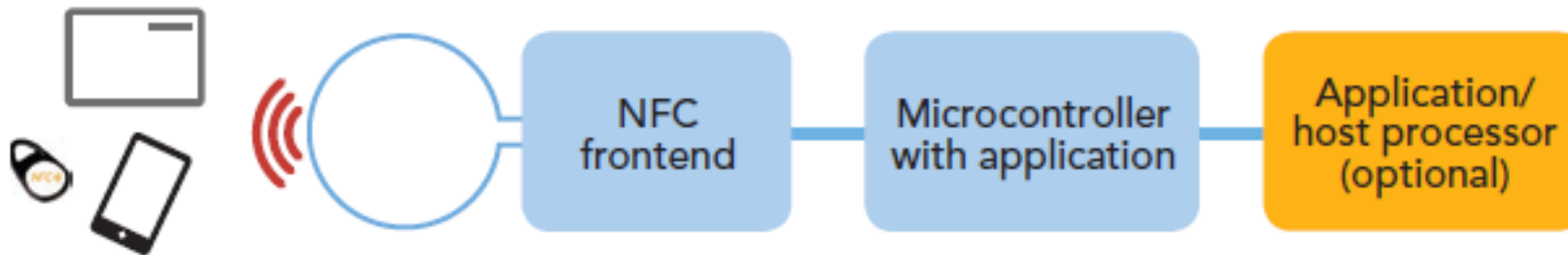
PN7120

OVERVIEW



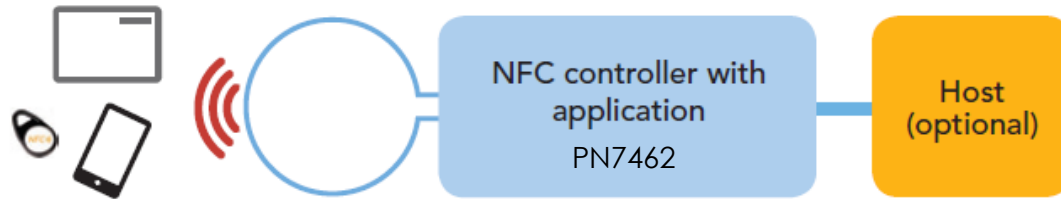
NFC Frontends

- NFC frontends are integrated transceiver ICs performing the contactless communication in charge of the modulation/demodulation, data processing and error detection
- Standalone NFC frontends are the most flexible way to add NFC to a system
- Easy to design in through dedicated development boards and SW libraries
- Broad product portfolio, from high output power reader to full NFC functionality

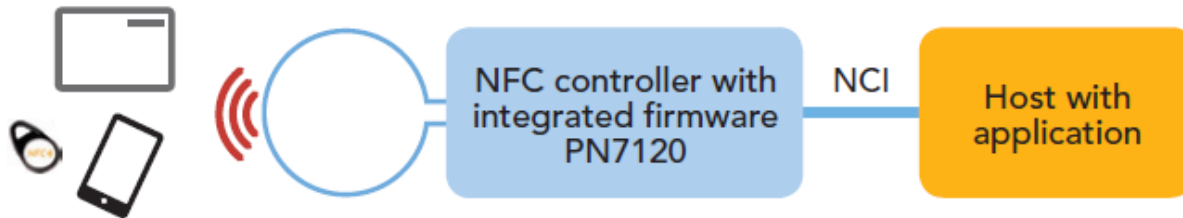


NFC Controllers

- NFC controller solutions enable higher integration with fewer components, since they combine an NFC frontend with an advanced 32-bit microcontroller.
- Two options:
 - Customizable FW: Ability to load fully-custom applications.



- Integrated FW: For an easy, standardized interface



PN7120

Customer Benefits

- Low PCB footprint
- Low power consumption
- EMVCo 2.3.1a PCD analog and digital
- NFC Forum Device Requirements v1.3
- Full SW stack available for integration within Linux and Android 4.4.x and 5.x

Features

Ease of integration

- Direct connection to 5.5V device battery
- Flexible clock supply concept
- Supports both 1.8 and 3V connections to host controller
- Buffered output drivers to connect an antenna with minimum number of external components

Flexibility in use case supports

- Fully configurable polling loop with low power modes for automated device discovery
- Autonomous mode when host is shut down (host can be in a deep sleep mode and be awakened via IRQ pin by PN7120 when entering RF field)

[pn7120 datasheet](#)

RF Communication Modes

Reader/Writer modes

- NFC Forum tags Type 1, 2, 3, 4 and 5
- ISO/IEC 14443 Type A & B, R/W up to 848 Kbps
- ISO/IEC 15693 Tags (ICODE)
- FeliCa tags up to 424 Kbps
- MIFARE 1K/4K
- MIFARE DESFire
- Kovio ink printed tags

Card modes

- ISO/IEC 14443-A and B card emulation via host

P2P modes

- Active and passive initiator and target according to ISO/IEC 18092 at all data rates (106 kbps to 424 kbps)

Interfaces

- I2C up to 3,4MBaud/s
- NFC Forum NCI 1.0 compliant protocol

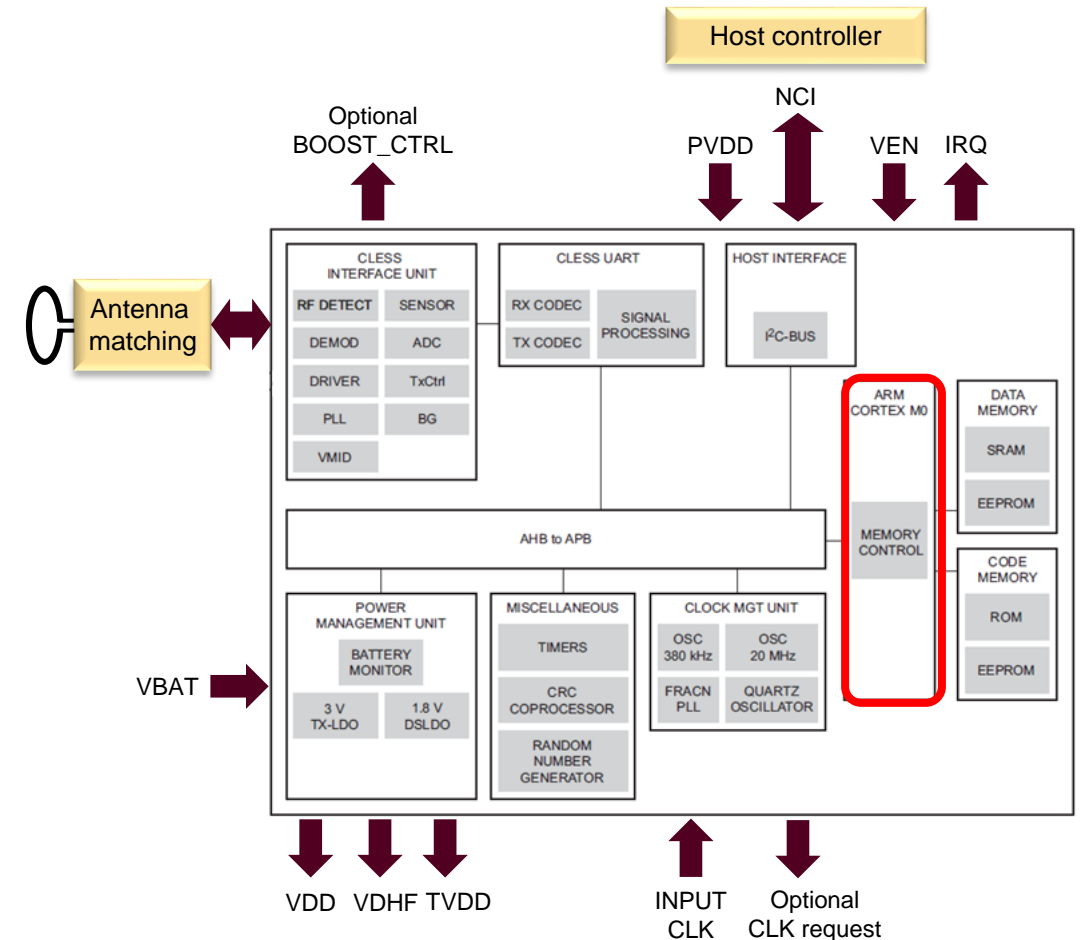
Package

- VFBGA49

PN7120

Block diagram

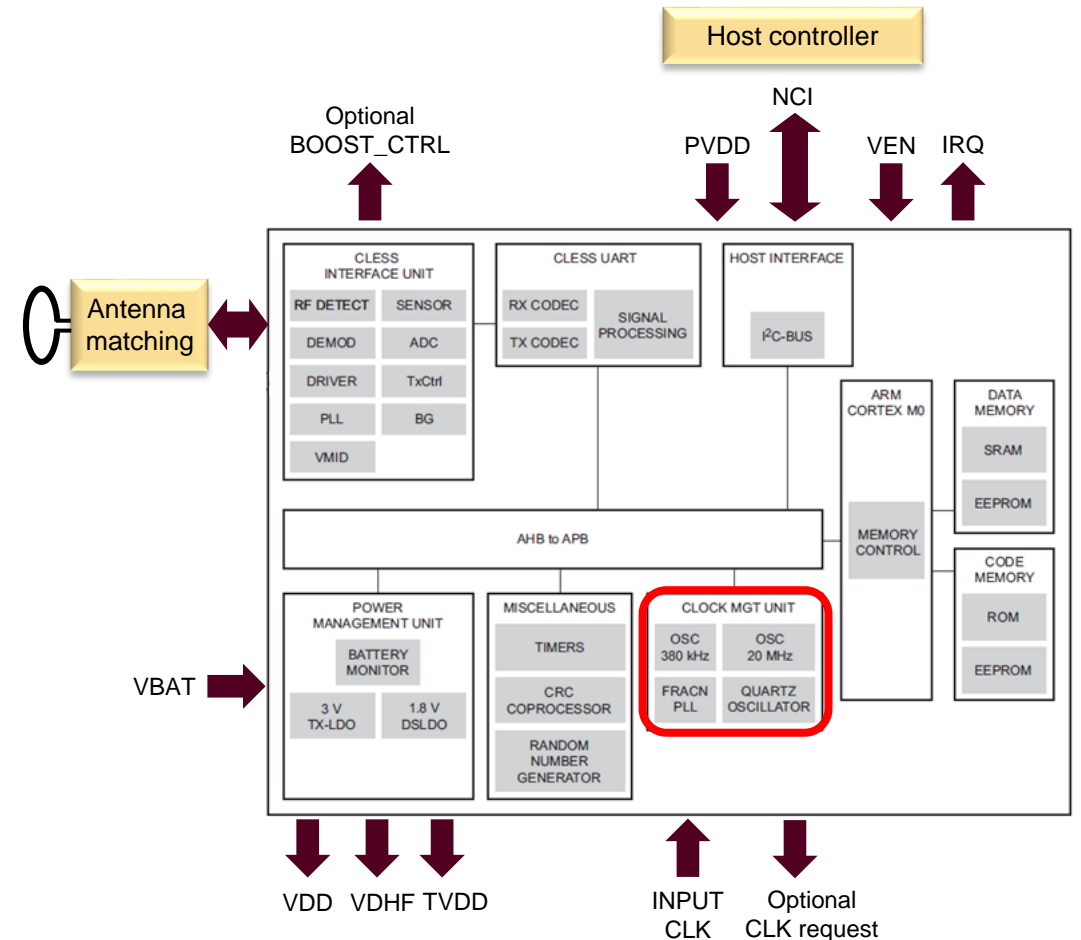
- ARM Cortex-M0 for integrated firmware
- Host interface
 - Link with host controller (NCI over I2C)
- Clock interface
 - Clock source required when generating the RF field
- Power interface
 - Interface to power management unit (direct battery supply supported)
- Antenna interface
 - Link to an NFC antenna in order to enable communication with a remote contactless device



PN7120

Clock management unit

- Use of crystal oscillator
 - A 27.12 MHz crystal can be used as input clock for PN7120
 - When using a crystal, frequency accuracy and drive level must be carefully selected according to the specs
- Use of system clock
 - Input clock frequency must be one of the following values: 13MHz, 19.2MHz, 24MHz, 26MHz, 38.4MHz or 52MHz.
 - Based on this input clock signal, the PN7120 integer PLL generates the required 27.12MHz internal clock for field generation.
- Clock request mechanism:
 - In order to optimize the device power consumption, the input clock could be provided by the system only when it is actually needed by the chip

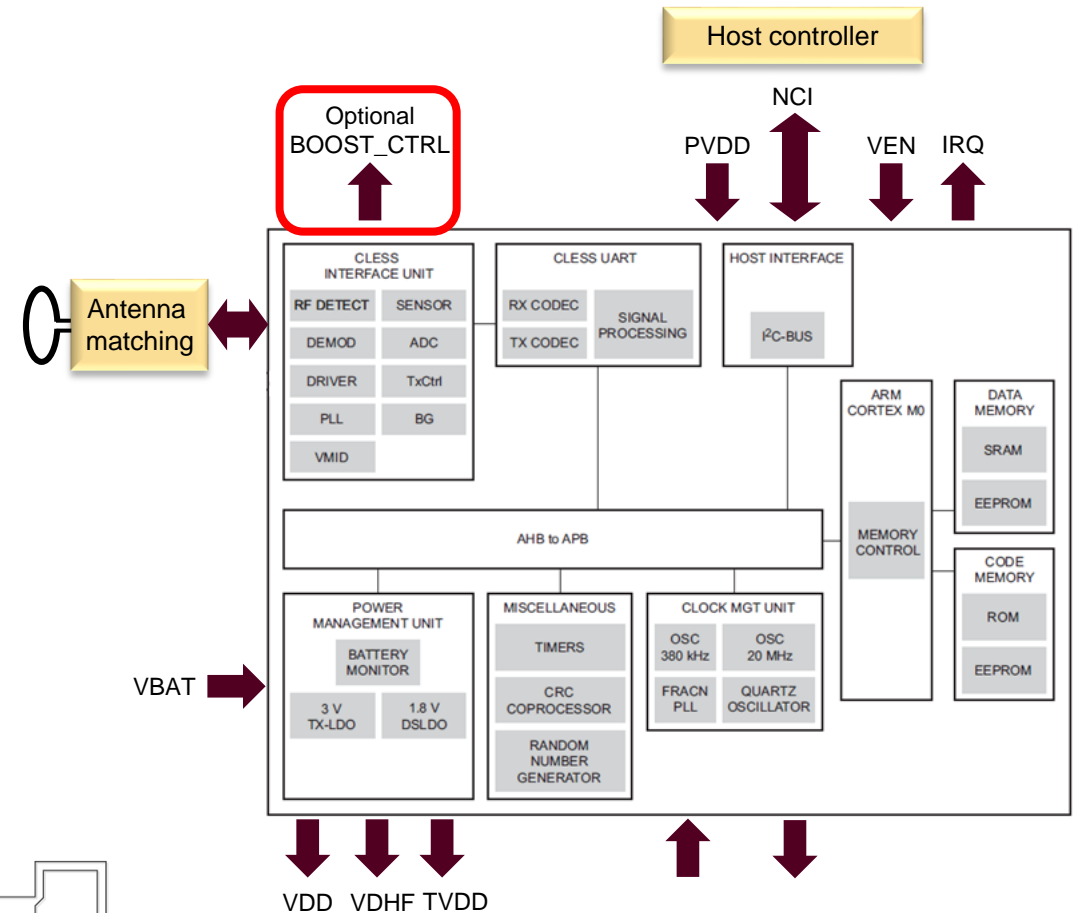
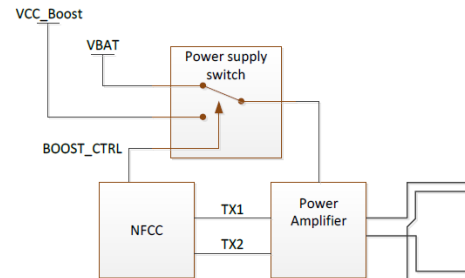


PN7120

Optional booster control

- An external booster can add to the hardware design to achieve better communication distance and RF performance.
- The PN7120 offers a control of the booster circuitry to optimize the overall power consumption
 - This is done via the PN7120 BOOST_CTRL pin

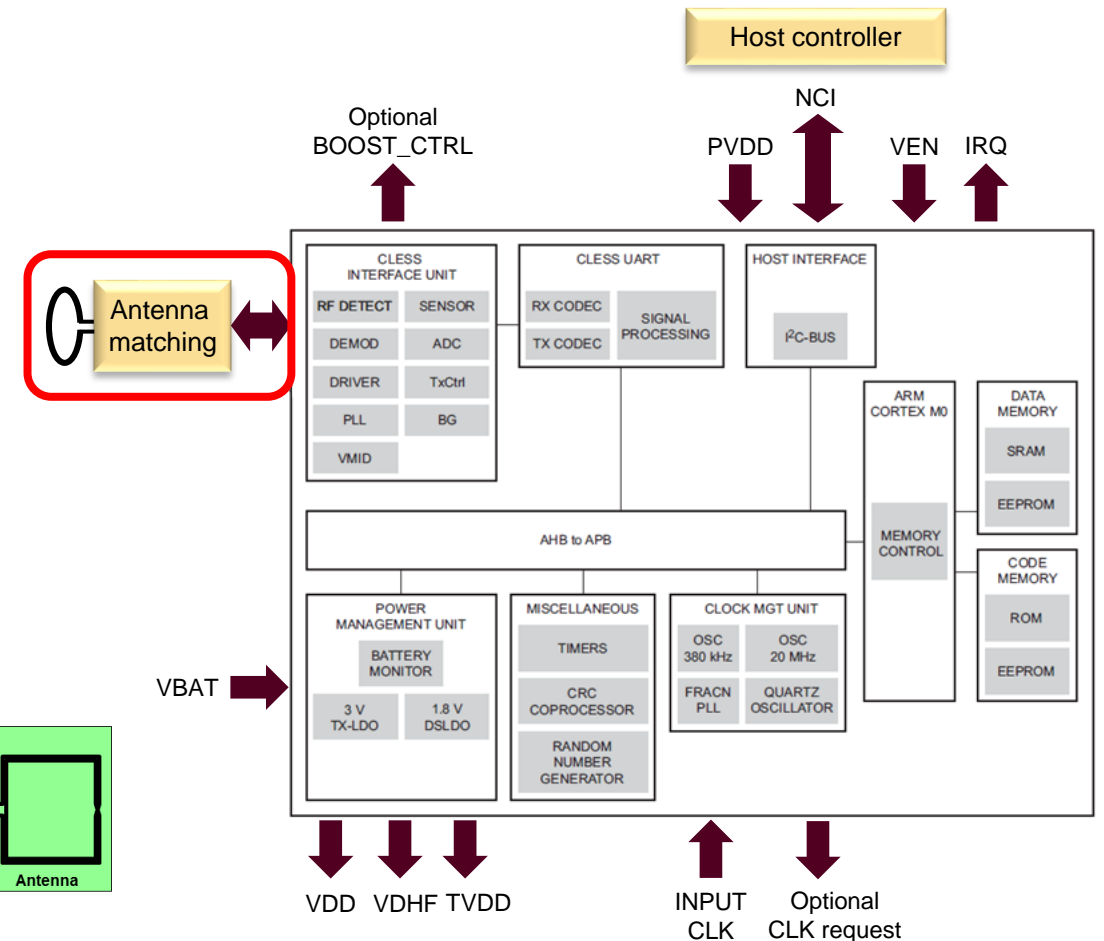
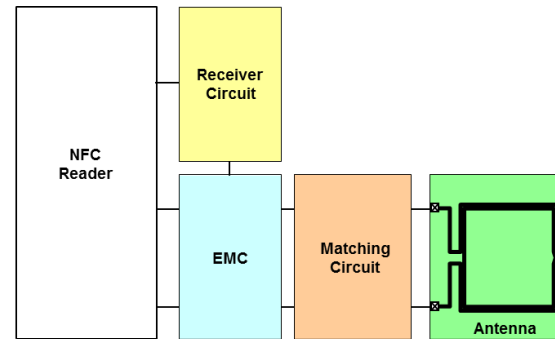
[more info: an11565 hardware design guide](#)



PN7120

Antenna matching block

- The PN7120 is intended to be connected to an external coil antenna through a specific matching/tuning network

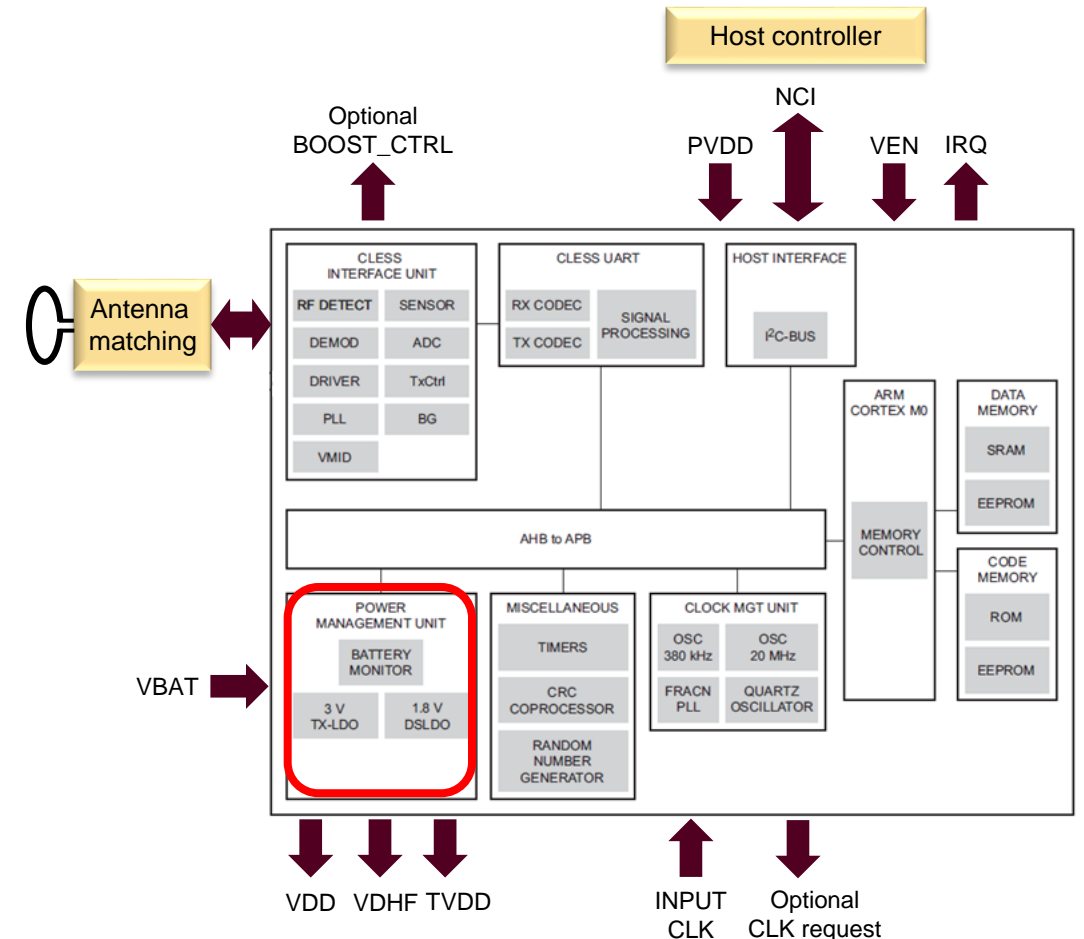


[More info: AN11564 PN7120 Antenna and tuning design guide](#)

PN7120

Power management unit

- Highly efficient integrated power management unit (PMU) allows direct supply from a battery
 - It is able to operate with a wide voltage input range from 5.5V down to 2.75V
- PN7120 is designed in order to enable different power states:
 - Monitor: The system is in power off since it is supplied with voltage below its programmable critical level
 - Hard power down: The system is in power off to have minimum power consumption
 - Standby: The system is kept supplied to enable configured wake-up sources which allow us to switch to Active state
 - Active: The system is on power on and PN7120 internal blocks are supplied.
- At application level, the PN7120 continuously switches between different states to optimize current consumption



PN7120

Low power discovery mode

- Average power consumption depends on:
 - Polling modes enabled by host controller
 - Listen mode duration (standby current)
 - Antenna system used by the application (RF impedance)
- PN7120 implements two additional modes of Low Power Polling:
 - Low power tag detector
 - Replaces each regular polling cycle by an RF pulse (few μ s)
 - It uses an enhanced HW reception path in order to reliably detect the insertion of a tag within the field.
 - NXP provides a proprietary extension to the NCI driver in order to enable and configure this mode
 - Hybrid mode
 - Introduces a regular polling cycle after a defined amount of LPCD pulses if a tag is still not detected

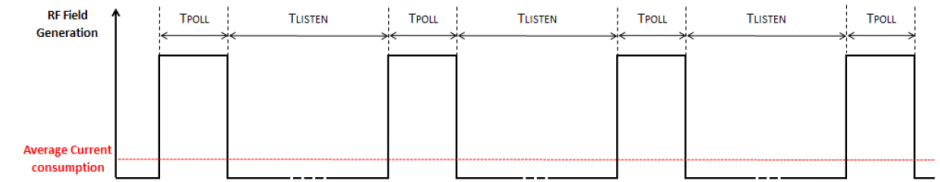


Fig 1. Regular polling loop

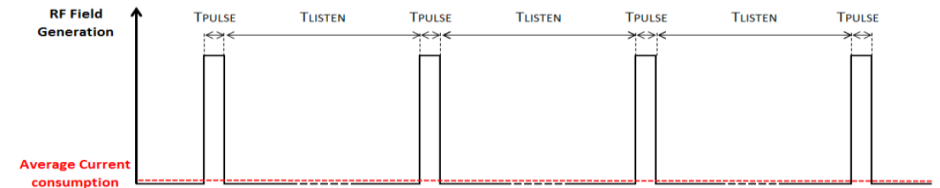


Fig 2. Low power card detector loop

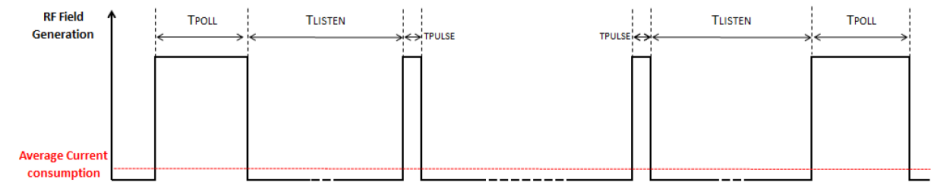


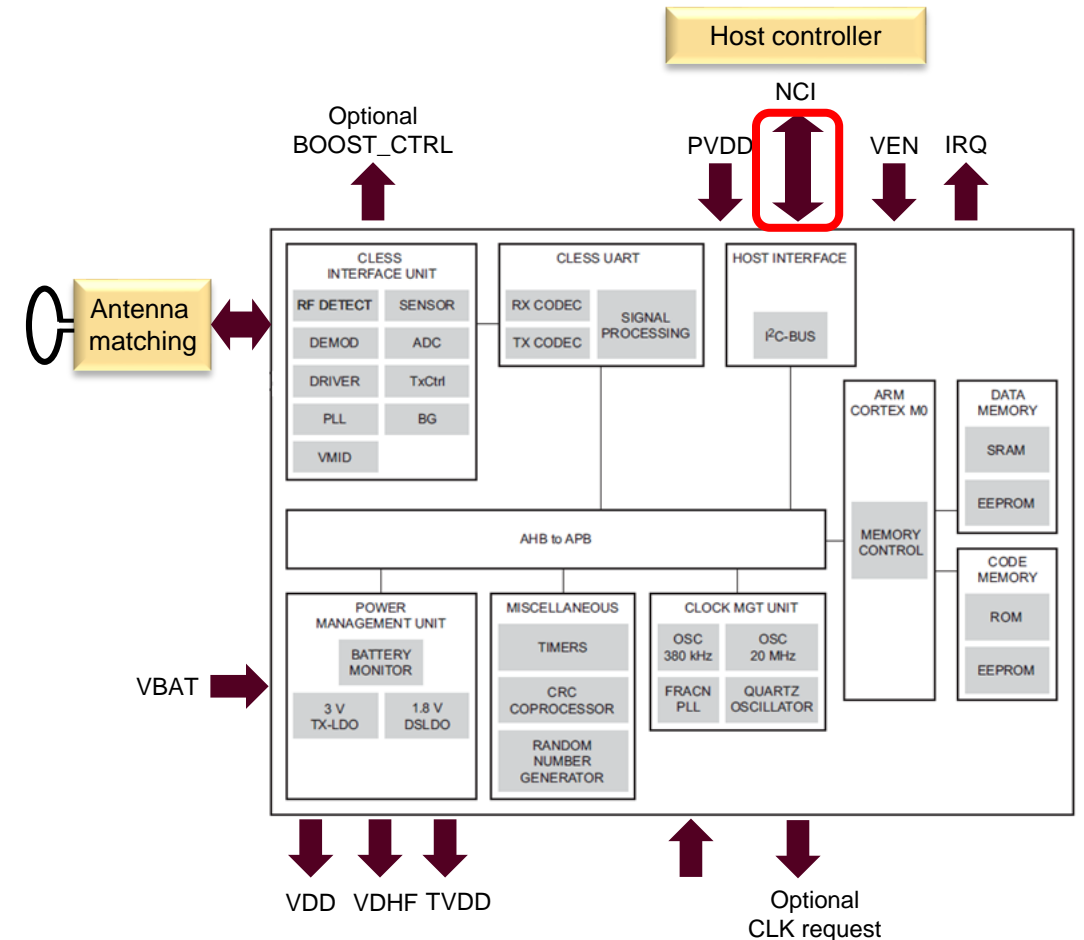
Fig 3. Hybrid polling loop

More info: [AN11562 PN7120 Low Power mode configuration](#)

PN7120

NCI interface

- The NCI specification defines a standard interface within an NFC device between an NFC controller and the device's main application processor
- The NCI interface provides manufacturers with a standard interface they can use for whatever kind of NFC-enabled device they build

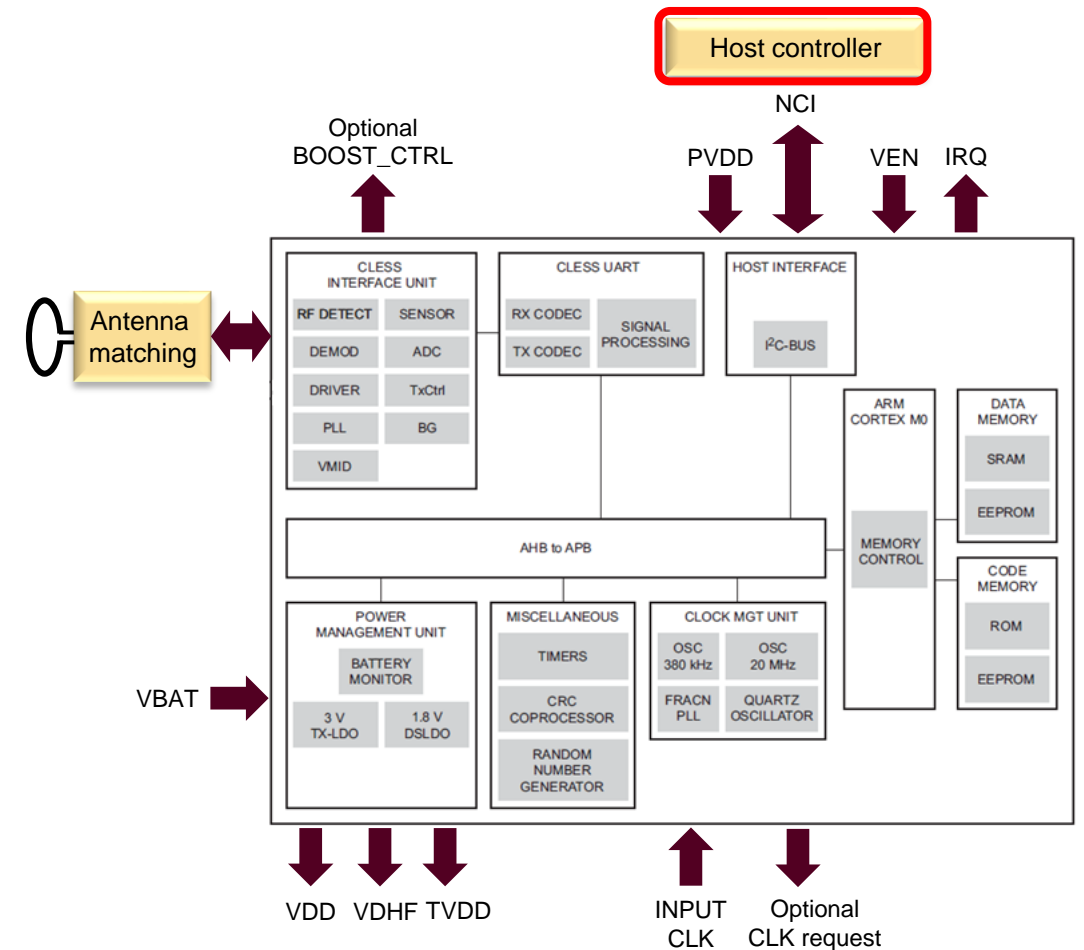


[More info: UM10819 - PN7120 User Manual](#)

PN7120

Host controller integration

- NCI stack is available in Android and Linux systems
- Additionally, NXP provides software extensions to make use of the full PN7120 capabilities



NFC Controller Interface (NCI)

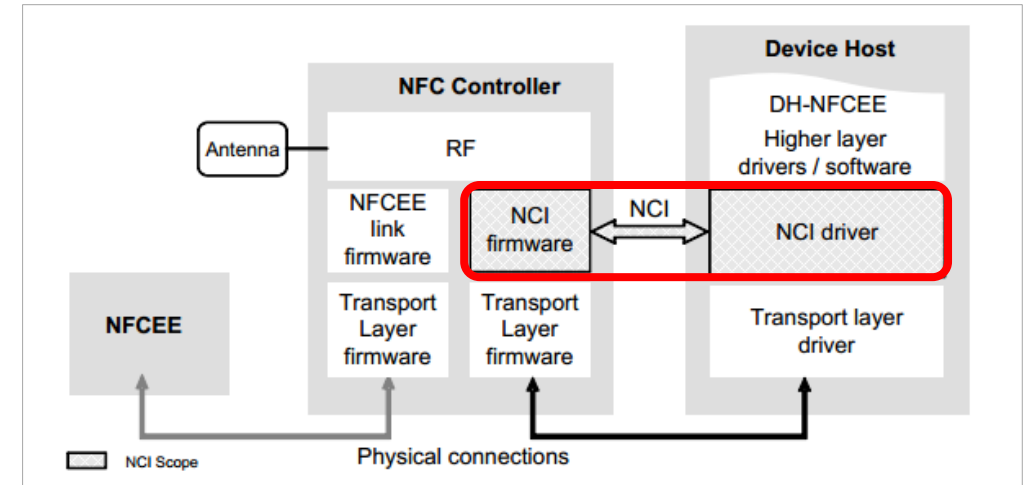
NFC Forum TS-NCI-1.1

Aim of the NCI specification:

- Defines an interface within the NFC device between an NFC Controller and the device's main application processor (DH)
- How it does this:
 - NCI offers users a logical interface that can be used with different physical transports, such as UART, SPI or I2C.

Implementation:

- Requires software implementation in both NFC Controller and device main application processor (linked with device running OS).
- Scope:
 - NCI scope is limited to the interface between the Device Host (DH) and the NFC controller (NFCC)



<http://nfc-forum.org/our-work/specifications-and-application-documents/specifications/nfc-forum-technical-specifications/>

PN7120

Support material

Doc ID	Doc Name	Description
939775017621	PN7120 Leaflet	Full NFC Forum-compliant controller with integrated FW and NCI interface
PN7120	PN7120 Product Data Sheet	Describes the functionality and electrical specification of the PN7120 NFC controller
AN11700	PN7120 Product Quick Start Guide	Describes the PN7120 documentation to be used to start working with PN7120
UM10819	PN7120 User Manual	Describes the PN7120 interfaces, modes of operation and possible configurations.
AN11565	PN7120 Hardware Design Guide	Provides an overview on how to integrate the PN7120 NFC controller from hardware perspective
AN11564	PN7120 Antenna and tuning Design Guide	Provides guidelines regarding the way to design and tune an NFC antenna for the PN7120
AN11562	PN7120 Low Power mode configuration	Provides guidance on how PN7120 can be configured in order to reduce current consumption by using low power poll mode
AN11658	NXP-NCI NulIOS integration example	Intended to provide a description of the NXP-NCI demo project demonstrating simple integration of NXP-NCI based NFC Controller without any OS support required.
SW324110	NXP-NCI LPCXPRESSO example project	
AN11697	PN7120 Linux SW stack integration guidelines	Provides guidelines for the integration of NXP's PN7120 NFC controller to a generic GNU/Linux platform from SW perspective
AN11690	NXP-NCI Android porting guidelines	Describes how to add support for a NXP NCI NFC chip to an Android based system.

product website pn7120:

www.nxp.com/products/identification_and_security/nfc_and_reader_ics/nfc_controller_solutions/pn7120a0ev.html

NFC Forum

A global, special interest group (SIG)

- The mission of the NFC Forum is to advance the use of NFC technology by:
 - Developing standards-based specifications that ensure interoperability among devices and services
 - Encouraging the development of products using NFC Forum specifications
 - Educating the market globally about NFC technology
 - Ensuring that products claiming NFC capabilities comply with NFC Forum specifications
 - Promoting the NFC Forum N-Mark
- NFC Forum approved 21 specifications spanning:
 - Data exchange formats
 - Tag types
 - Record type definition
 - Device interface controller → NCI
 - Protocols
 - Reference applications
 - Personal Health Device Communications



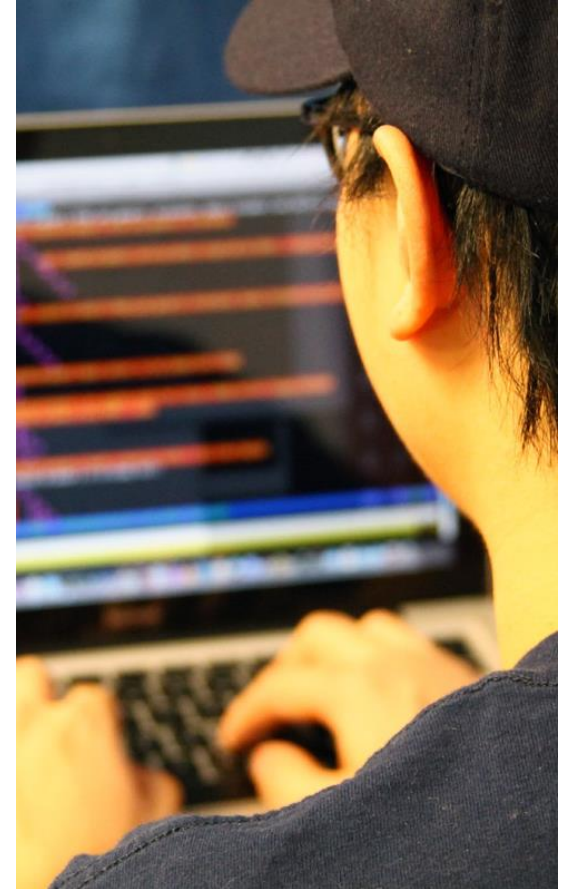
LINUX, ANDROID AND RTOS NFC STACKS



NFC in Linux

Way forward to implement your stack

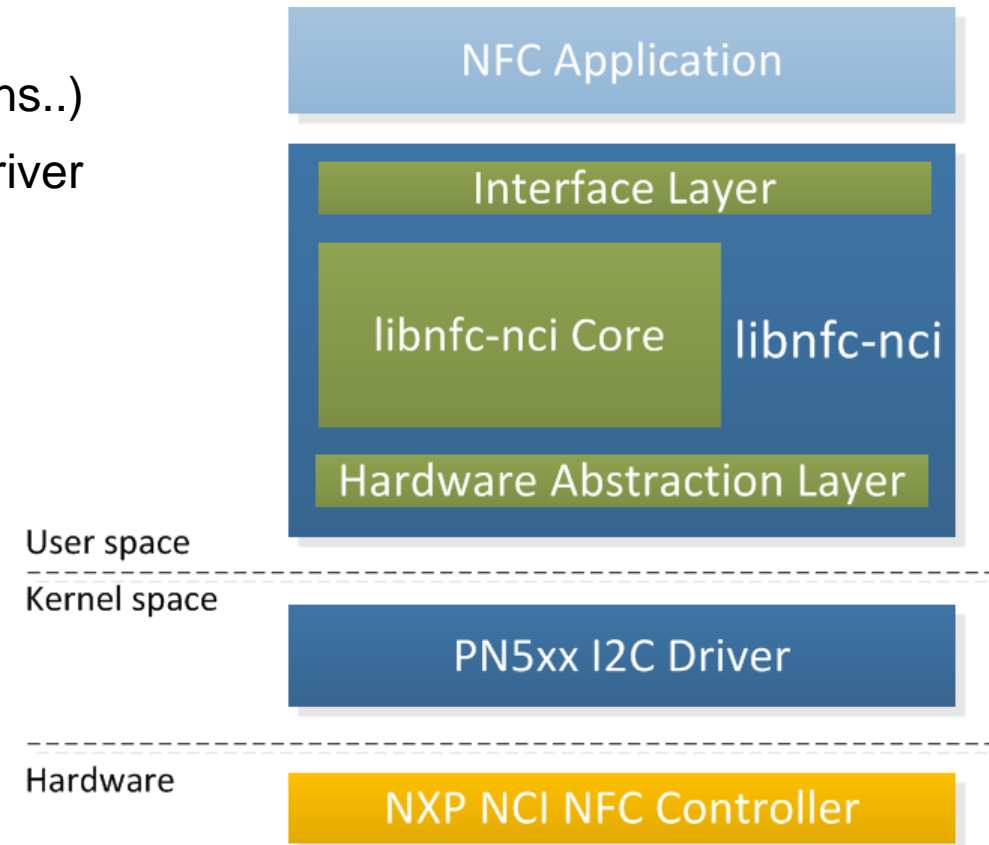
- **Option 1: Use Linux open source stack**
 - The stack will be maintained by the community
 - NXP will not provide support. No more contribution from NXP to the stack in kernel.org.
 - For its NCI interface based products NXP participated in improvements of this Kernel based stack (v1.2.0 being up streamed into kernel.org)
- **Option 2: Use Linux libnfc-nci stack from NXP (recommended)**
 - NXP's SW product "Linux libnfc-nci stack" is derived from our available and proven Android stack
 - From June 2015 onwards NXP will provide and maintain the NFC driver in user mode and will be distributed through GitHub:
 - https://github.com/NXPnfcLinux/linux_libnfc-nci
 - Install PN5xx_I2C kernel driver (without dependencies to kernel version)
 - Install Linux libnfc-nci stack (full user mode)
 - Port existing application on the Linux libnfc-nci stack (similar to nearDAL)



PN7120 SW Integration in Linux OS

Linux libnfc-nci stack: Architecture overview

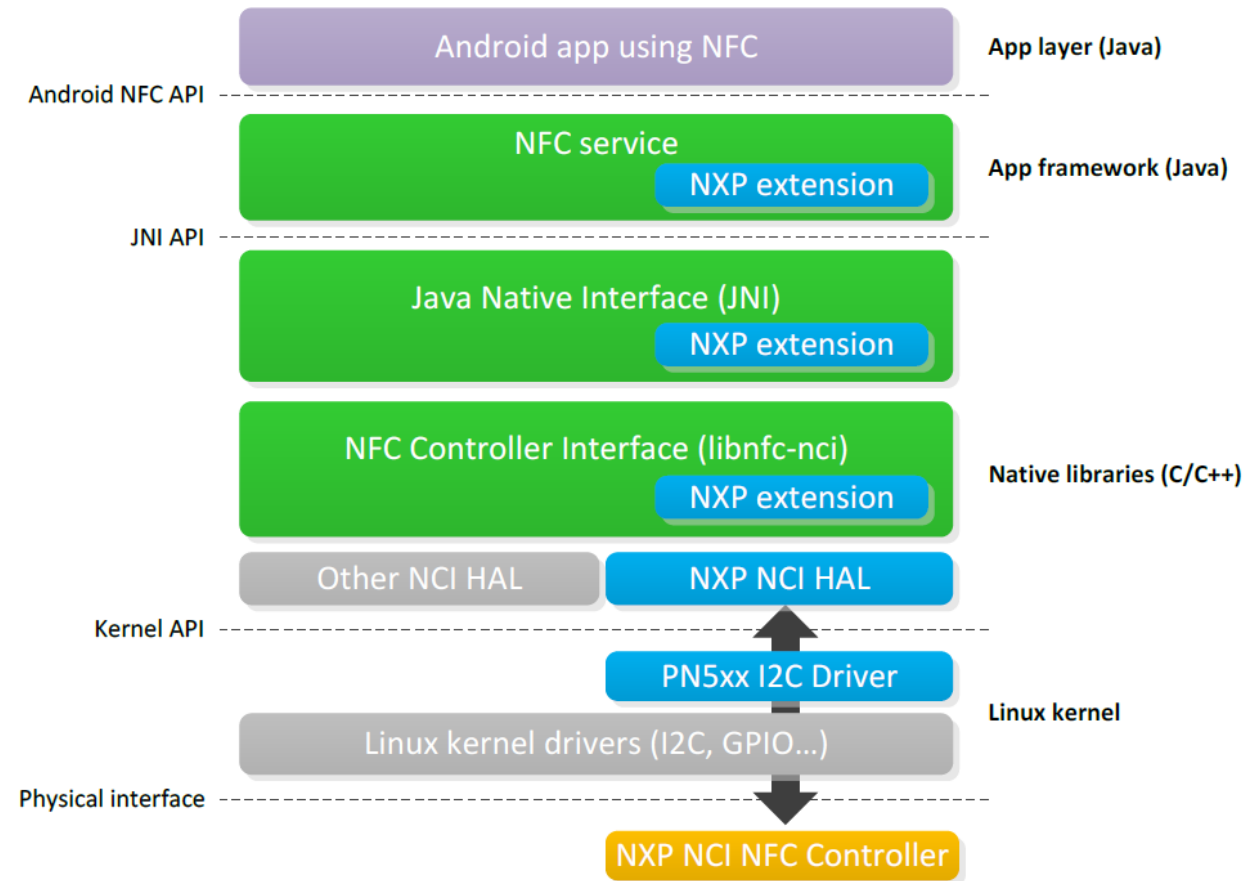
- **Interface Layer:** exposes the library API
- **Core Layer:** implements NFC features (NCI, NDEF, Tag operations..)
- **Hardware abstraction layer:** provides connection to the kernel driver as well as basic functionalities like self-test or FW update
- List of features supported by Linux libnfc-nci stack:
 - NDEF tag support
 - MIFARE classic tag support
 - P2P, LLCP, SNEP
 - Wi-Fi & Bluetooth handover
 - Raw tag command support
 - Proprietary NCI command support
 - Host Card Emulation



PN7120 SW integration in Android

Android NFC stack

- **NFC service:** API within the Android framework that provides access to the NFC functionality
- **JNI:** Glue code between Java classes and Native classes (written in C/ C++)
- **NXP NCI HAL:** NXP hardware specific implementation supporting full capabilities
- **Linux Kernel:** Implements the hardware drivers and speaks to the underlying NFC controller



PN7120 SW Integration in RTOS or without OS Systems

e.g.: LPCXpresso platform and Kinetis Design Studio

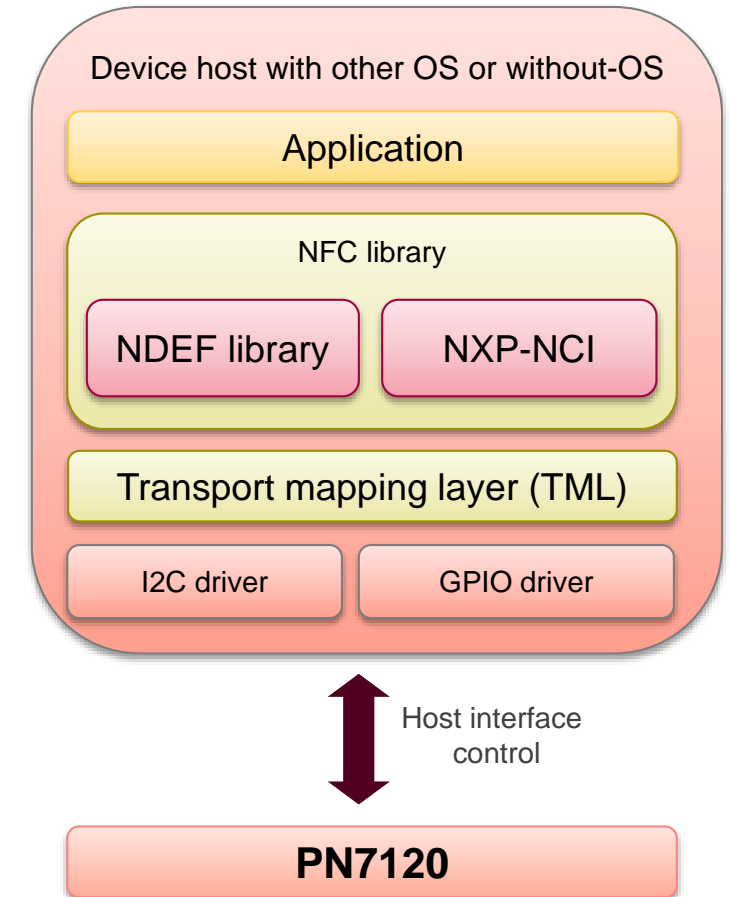
- PN7120 NCI based NXP NFC Controller can be integrated to an embedded system with no OS resources in order to provide NFC capability
- There is a project example based on LPCXpresso platform, running on NXP LPC122x or LPC11xx MCUs driving the NFC controller
- This example can easily be ported in a system with RTOS support.

Architecture overview:

- *NXP-NCI*: High level NFC API for the connection and configuration of the NFC controller
- *NDEF library*: Submodule handling NDEF functionality for R/W, P2P and CE modes.
- *TML*: Brings HW abstraction to the NFC library
- *I2C driver*: Handler for the I2C communication with the NFC controller
- *GPIO driver*: handler for the NFC controller VEN (Hard reset) and IRQ (interface interrupt) pins.

Easy porting to other platforms:

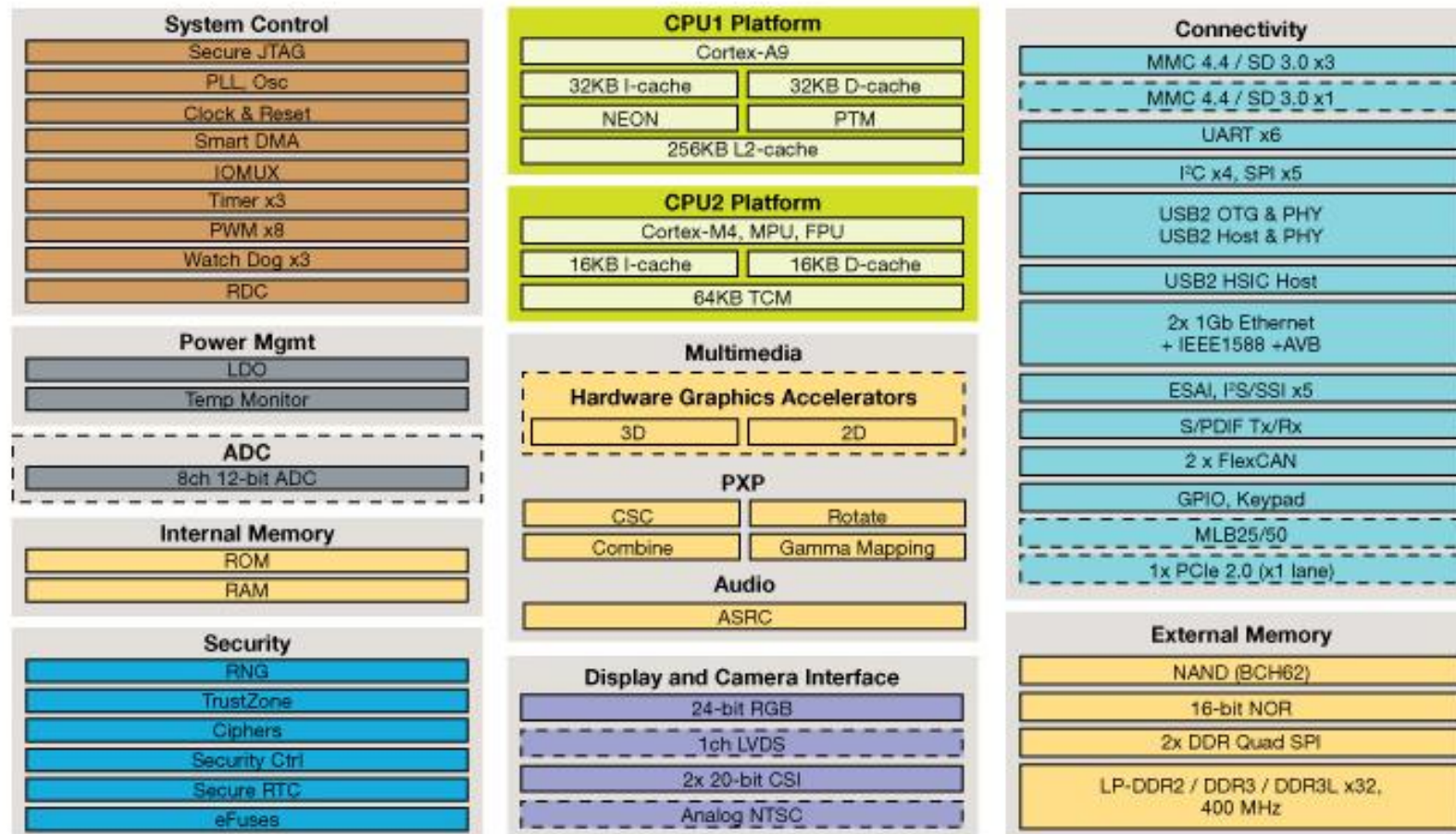
- *NFC Library* and *TML* layers can be reused
- Only *I2C driver* (3 functions) and *GPIO driver* (4 functions) need to be adapted to each system



i.MX 6SoloX AND LAB HARDWARE HIGH LEVEL OVERVIEW



i.MX 6SoloX



[.MX6SX: i.MX 6SoloX Processors](#)

i.MX 6 SoloX Advantages

- **Heterogeneous Architecture with Smart System Power**

- Single Cortex-A9 paired with a Cortex-M4
- Enables concurrent execution of multiple software environments to provide high performance with real time responsiveness, allowing for smart system power.

- **Optimized Integration for Design Flexibility**

- Dual Gb Ethernet with hardware AVB support for fast reliable communication
- PCIe for high-speed connectivity (e.g. Wi-Fi)
- 2D and 3D hardware graphics acceleration for performance optimized UI
- Memory controller supports low-power LPDDR2 and cost-effective DDR3/DDR3L

- **Optimized Power**

- Maintain a system aware and power efficient state with complete shut down of the Cortex-A9 core, with the Cortex-M4 still active and performing low-level system monitoring tasks.

- **Secure Solutions for Optimized Performance Efficiency**

- On-chip resource domain controller providing a centralized programming model to configure isolation and sharing of system resources.
- Advanced security supporting high assurance (secure) boot, cryptographic cipher engines and random number generator.

UDOO Neo Extended

- Full Specs
- Freescale™ i.MX 6SoloX applications processor with an embedded ARM Cortex-A9 core and a Cortex-M4 Core
- Integrated 2d/3d graphics controller
- RAM 1GB
- Micro HDMI interface - LVDS interface + touch (I2C signals)
- Analog camera connection supporting NTSC and PAL - 8-bit parallel camera interface
- MicroSD card slot onboard - 8-bit SDIO interface
- HDMI audio transmitter - S/PIDF & I2S
- 1x USB 2.0 Type A ports - 1x USB OTG (micro-AB connector)
- Wi-Fi 802.11 b/g/n, Direct Mode SmartConfig and Bluetooth 4.0 Low Energy
- 3 x UART ports – 2 x CAN Bus interfaces
- 8 x PWM signals – 3 x I2C interface – 1 x SPI interface – 6 x multiplexable signals
- Power Supply: 5 V DC Micro USB, 6-15 V DC Power Jack, RTC Battery Connector
- Green Power Status LED - User Configurable LEDs (Red and Orange)
- Integrated Sensors: 3-Axis Accelerometer - 3-Axis Magnetometer - 3-Axis Digital Gyroscope - 1x Sensor Snap-In I2C connector
- Dimensions: 89mm x59mm (3.50" x 2.32")
- Arduino-Compatible through the standard Arduino Pins layout and compatible with Arduino shields
- 32 extended GPIOs (A9 dedicated) - 22 Arduino GPIOs (M4 dedicated)
- 6 available Pins
- Android Lollipop & Linux UDObuntu2 (14.04 LTS)

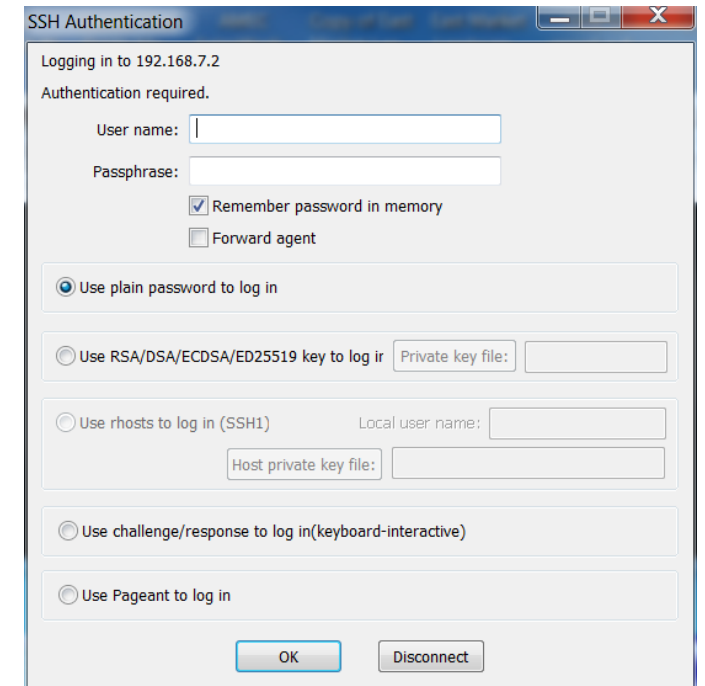
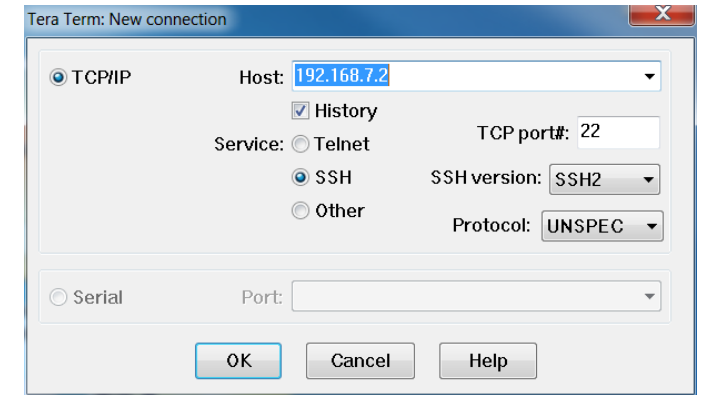


NFC COMMISSIONING LAB



Setup

1. Plug the USB connector of your udoo board into the PC
2. Wait ~ 1 min for the board to power up
3. A window may pop up that says “Do you want to scan and fix “boot”
4. Click “**Continue without scanning**”
5. Double click on the Tera Term icon
6. Type in 192.168.7.2 in the “Host” field and check the “TCP/IP” and “SSH” radio buttons if they are not already checked.
7. Ignore the security warning window and click continue
8. The User’s name and Password are the same. Type in “udooer”
9. You should see a terminal window appear. Stretch the window to make it larger
10. Now we are ready to start the demos!



Pairing to a Bluetooth Speaker, Finding the MAC

1. Go to “Settings under “Wireless & Networking” tap on “More...” and ensure your NFC is turned on.
2. Ensure the Bluetooth speaker is OFF and your phones Bluetooth is OFF!
3. In the terminal window type in the command “nfcDemoApp poll”.
4. You should see the below appear in the terminal.

```
#####  
##                      NFC demo                      ##  
#####  
##                      Poll mode activated              ##  
#####  
... press enter to quit ...
```

Waiting for a Tag/Device...

5. With the speaker powered off place and then remove (TAP) the top of the speaker over the antenna, the red dashed line shows the antenna location.

Look over the various fields.

- a. What type of NFC tag is inside the speaker? Type A
 - b. What is the UID? 04:28:EB:31:00:50:99 (something like this)
 - c. What is the max NDFE record size? 142 Bytes
 - c. How much memory is used (what is the message size)? 58 bytes
6. Go to the next few pages and see a more detailed break down of what the tag has stored in it. DON'T CLEAR YOUR SCREEN!



Continued

- ** TagInfo scan (version 3.0) 2016-04-22 10:54:20 **
- -- INFO -----
- # IC manufacturer:
NXP Semiconductors
- # IC type:
NTAG203(F)
- # NFC Forum NDEF-compliant tag:
Type 2 Tag
- -- NDEF -----
- # NFC data set information:
NDEF message containing 1 record
Current message size: 58 bytes
Maximum message size: 142 bytes
NFC data set access: Read & Write
Can be made Read-Only
- # Record #1: Bluetooth Secure Simple Pairing record:
Type Name Format: MIME type (RFC 2046)
Short Record, ID Length present
ID: "0"
type: "application/vnd.bluetooth.ep.oob"
OOB data length: 21 bytes
MAC address: FC:58:FA:25:82:A4
* Unknown manufacturer
Complete local name: "Anker A7910"
Payload length: 21 bytes
Payload data:

[00] 15 00 A4 82 25 FA 58 FC 0C 09 41 6E 6B 65 72 20
|....%.X...Anker |
[10] 41 37 39 31
30 |A7910 |

```
# NDEF message:
[00] DA 20 15 01 61 70 70 6C 69 63 61 74 69 6F 6E 2F |. ..application/|
[10] 76 6E 64 2E 62 6C 75 65 74 6F 6F 74 68 2E 65 70 |vnd.bluetooth.ep|
[20] 2E 6F 6F 62 30 15 00 A4 82 25 FA 58 FC 0C 09 41 |.oob0....%.X...A|
[30] 6E 6B 65 72 20 41 37 39 31 30 |nker A7910 |

# NDEF Capability Container (CC):
Mapping version: 1.0
Maximum NDEF data size: 144 bytes
NDEF access: Read & Write
E1 10 12 00 |.... |

-- EXTRA -----
# Memory size:
168 bytes total memory
* 42 pages, with 4 bytes per page
* 144 bytes user memory (36 pages)
# IC detailed information:
Full product name:
* NT2H0301G0DUD or NT2H0301F0DTx

-- TECH -----
# Technologies supported:
ISO/IEC 14443-3 (Type A) compatible
ISO/IEC 14443-2 (Type A) compatible
# Android technology information:
Tag description:
* TAG: Tech [android.nfc.tech.MifareUltralight, android.nfc.tech.NfcA, android.nfc.tech.Ndef]
android.nfc.tech.Ndef
android.nfc.tech.MifareUltralight
android.nfc.tech.NfcA
* Maximum transceive length: 253 bytes
* Default maximum transceive time-out: 24576 ms
# Detailed protocol information:
ID: 04:28:EB:31:00:50:99
ATQA: 0x4400
SAK: 0x00
```



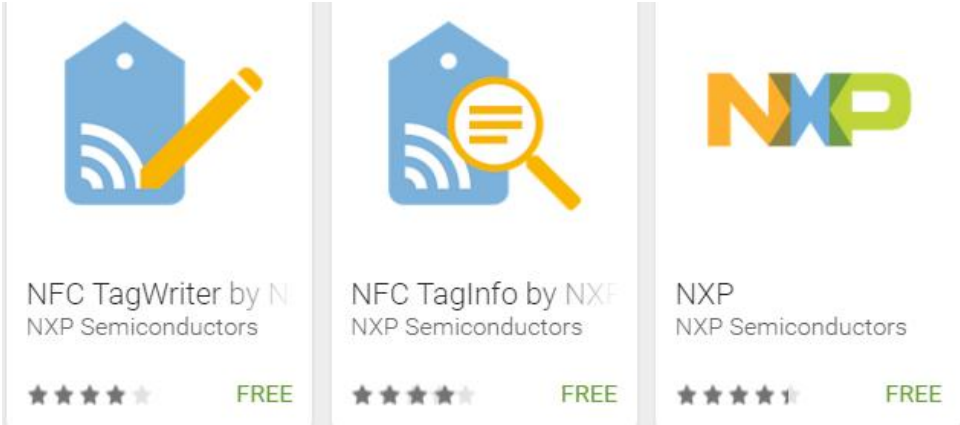
Continued

```
• # Memory content:
[00] * 04:28:EB 4F (UID0-UID2, BCC0)
[01] * 31:00:50:99 (UID3-UID6)
[02] . F8 48 00 00 (BCC1, INT, LOCK0-LOCK1)
[03] . E1:10:12:00 (OTP0-OTP3)
[04] . 03 3A DA 20 |... |
[05] . 15 01 61 70 |..ap|
[06] . 70 6C 69 63 |plic|
[07] . 61 74 69 6F |atio|
[08] . 6E 2F 76 6E |n/vn|
[09] . 64 2E 62 6C |d.bl|
[0A] . 75 65 74 6F |ueto|
[0B] . 6F 74 68 2E |oth.|
[0C] . 65 70 2E 6F |ep.o|
[0D] . 6F 62 30 15 |ob0.|
[0E] . 00 A4 82 25 |...%|
[0F] . FA 58 FC 0C |.X..|
[10] . 09 41 6E 6B |.Ank|
[11] . 65 72 20 41 |er A|
[12] . 37 39 31 30 |7910|
[13] . FE 00 00 00 |....|
[14] . 00 00 00 00 |....|
[15] . 00 00 00 00 |....|
[16] . 00 00 00 00 |....|
[17] . 00 00 00 00 |....|
[18] . 00 00 00 00 |....|
[19] . 00 00 00 00 |....|
[1A] . 00 00 00 00 |....|
[1B] . 00 00 00 00 |....|
[1C] . 00 00 00 00 |....|
[1D] . 00 00 00 00 |....|
[1E] . 00 00 00 00 |....|
[1F] . 00 00 00 00 |....|
[20] . 00 00 00 00 |....|
[21] . 00 00 00 00 |....|
```

```
•
[22] . 00 00 00 00 |....|
[23] . 00 00 00 00 |....|
[24] . 00 00 00 00 |....|
[25] . 00 00 00 00 |....|
[26] . 00 00 00 00 |....|
[27] . 00 00 00 00 |....|
[28] . 00 00 -- -- (LOCK2-LOCK3)
[29] . 00 00 -- -- (CNT0-CNT1, value: 0)

• *:locked & blocked, x:locked,
  +:blocked, .:un(b)locked

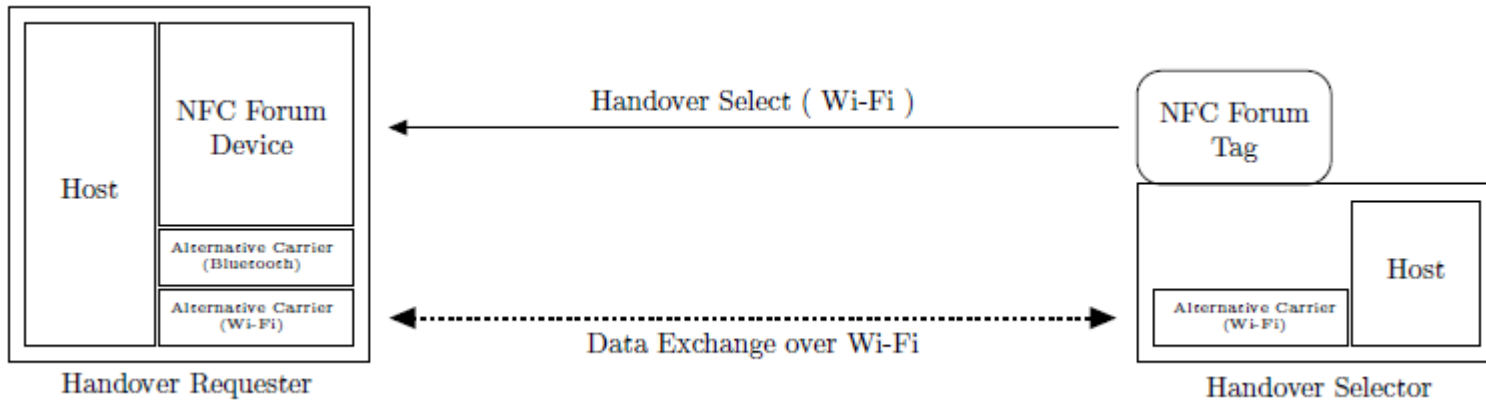
• -----
```



[google play store \(nxp nfc tools\)](#)



What is a static handover?



- A static handover can be used when the Handover Selector device does not constitute an NFC Forum Device but has a (cheaper) NFC Forum Tag attached.
- NFC Forum Tags provide storage space that can be read or written but might not have an internal connection with a Host/MCU.
- It is not possible for an NFC Forum Tag to receive and interpret a Handover Request Message and to dynamically construct a corresponding Handover Select Message.
- More sophisticated devices use Mediated Handovers like a PC connecting with a wireless printer



Connection Handover

Technical Specification

Version 1.3

2014-01-16

NFC Forum™

[CH]

Pair the Bluetooth Speaker to Your Android Phone (Static Handover)

1. Press `ENTER` in the terminal to stop the program
2. Find you BT MAC address in the NDEF record from the previous output. Start counting at byte 0, the MAC address begins at byte 39 and ends at byte 44, it is in Little Endian order! Write this address down.

3. Type in `"handOverDemo --bt -addr YOUR MAC -name ANKER A7910"`

Example command line followed by the demo response.

```
udooer@udooneo:~$ handOverDemo --bt -addr FC:58:FA:25:82:A4 -name ANKER A7910  
device name=ANKER A7910  
mac address=FC:58:FA:25:82:A4
```

```
...Waiting for a Tag or P2P device ...
```

4. Turn your Bluetooth Speaker ON and ensure your phones Bluetooth is ON.
5. Tap the phone on the demo boards antenna. You may hear the speaker beep and you will be asked if you want to pair with this device. Tap yes.
6. You phone should immediately connect to the speaker without tapping your phone to the speaker.
7. Play an audio file on your phone if you like.
8. Turn off the Bluetooth speaker.

What is the format of Bluetooth Configuration Message we created?

Table 12: Binary Content of a Sample Bluetooth OOB Data on an NFC Forum Tag

Offset (Octets)	Content	Length (Octets)	Explanation
0	0xD2	1	NDEF Record Header: MB=1b, ME=1b, CF=0b, SR=1b, IL=0b, TNF=010b
1	0x20	1	Record Type Length: 32 octets
2	0x21	1	Payload Length: 33 octets
3	0x61 0x70 0x70 0x6C 0x69 0x63 0x61 0x74 0x69 0x6F 0x6E 0x2F 0x76 0x6E 0x64 0x2E 0x62 0x6C 0x75 0x65 0x74 0x6F 0x6F 0x74 0x68 0x2E 0x65 0x70 0x2E 0x6F 0x6F 0x62	32	Record Type Name: application/vnd.bluetooth.ep.oob
35	0x21 0x00	2	OOB Optional Data Length (33 octets)
37	0x06 0x05 0x04 0x03 0x02 0x01	6	Bluetooth Device Address: 01:02:03:04:05:06
43	0x0D	1	EIR Data Length: 13 octets
44	0x09	1	EIR Data Type: Complete Local Name
45	0x48 0x65 0x61 0x64 0x53 0x65 0x74 0x20 0x4E 0x61, 0x6D 0x65	12	Bluetooth Local Name HeadSet Name
57	0x04	1	EIR Data Length: 4 octets
58	0x0D	1	EIR Data Type: Class of Device
59	0x04 0x04 0x20	3	Class of Device: <ul style="list-style-type: none"> 0x20: Service class = Audio 0x04: Major Device class = Audio/Video 0x04: Minor Device class = Wearable Headset Device
62	0x05	1	EIR Data Length: 5 octets
63	0x03	1	EIR Data Type: 16-bit Service Class UUID list (complete)
64	0x1E 0x11 0x0B 0x11	4	16-bit Service Class UUID list (complete): 0x111E – HFP-HF 0x110B – A2DP-SNK



Bluetooth® Secure Simple Pairing Using NFC

Application Document

NFC Forum™

NFCForum-AD-BTSSP_1_1

2014-01-09



Here is the code that built that message

```
bool LibNfcManager::writeNdefBluetoothMessage(int mTagHandle,unsigned char* pMacAddress,unsigned char* pDeviceName)
{
    unsigned char* lBluetoothMessage = (unsigned char*)malloc(sizeof(unsigned char)*255);
    int lBluetoothMessageLength,ret = 0;
    createBluetoothMessage(pMacAddress,pDeviceName,lBluetoothMessage,&lBluetoothMessageLength);
    ret = nfcTag_writeNdef(mTagHandle, lBluetoothMessage, lBluetoothMessageLength);
    free (lBluetoothMessage);
    if(ret == 0)
        return true;
    else
        return false;
}

void LibNfcManager::createBluetoothMessage(unsigned char* pMacAddress,unsigned char* pDeviceName,unsigned char* pMsg,int* pMsgLength)
{
    int currentIndex =0;

    unsigned char header = 0xD2;
    const char *type = "application/vnd.bluetooth.ep.oob";
    size_t typeLength = sizeof(unsigned char)*strlen((char*)type);
    size_t macAddressLength = sizeof(unsigned char)*strlen((char*)pMacAddress);

    //According to the bluetooth spec address is reversed.
    unsigned char* reverseMacAddress = (unsigned char*)malloc(macAddressLength * sizeof(unsigned char));
    reverseTab(pMacAddress,reverseMacAddress);
    size_t deviceNameLength = sizeof(unsigned char)*strlen((char*)pDeviceName);

    unsigned char bluetoothEIRDeviceName[] = {(unsigned char)deviceNameLength + 1,0x09};
    size_t payloadLength = (unsigned char) (deviceNameLength + sizeof(bluetoothEIRDeviceName) + macAddressLength + 2);
    *pMsgLength = payloadLength + typeLength + 3;

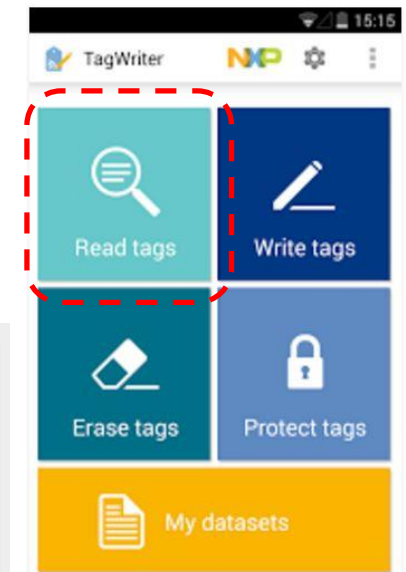
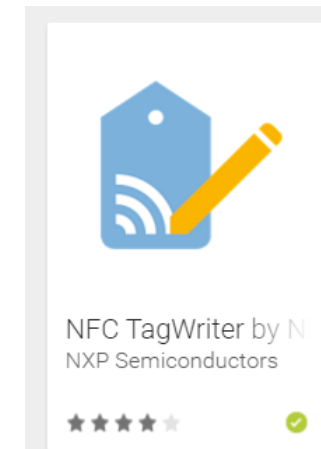
    //build message
    pMsg[currentIndex] = header;
    currentIndex += 1;
    pMsg[currentIndex] = typeLength;
    currentIndex += 1;
    pMsg[currentIndex] = payloadLength;
    currentIndex += 1;
    memcpy(&pMsg[currentIndex],type,typeLength);
    currentIndex += typeLength;
    pMsg[currentIndex] = payloadLength;
    currentIndex += 1;
    pMsg[currentIndex] = 0x00;
    currentIndex += 1;
    memcpy(&pMsg[currentIndex],reverseMacAddress,macAddressLength);
    currentIndex += macAddressLength;
    memcpy(&pMsg[currentIndex],bluetoothEIRDeviceName,2);
    currentIndex += 2;
    memcpy(&pMsg[currentIndex],pDeviceName,deviceNameLength);

    return;
}
```

Pairing a Device to a WI-FI Network

Android Version Kit Kat 4.4.X

1. Determine what version of Android you have. Go to “Settings” on your phone and scroll down to “About Phone” and look for the Android Version, if it is 4.4.2 you have Kit Kat. If it is lower than that let me know. If it is 5.X or greater you have Lollipop (SKIP TO THE NEXT PAGE)
2. Go to the “Google Play Store” and download, install and open the “NXP TagWrite App”
3. Do a Ctrl-C in the terminal to stop the program.
4. Type `handOverDemo --help` look over the flags. Note the double dashes and the single dashes in the command!
5. I will provide the below settings delineated by “???”.
6. Type in `handOverDemo --wifi -ssid NXP-HOW -pwd nxp1234567 -auth WPA2PSK -enc AES`
7. Place the provided card over the Reader Antenna to write the data into the card.
8. You should see `Write Wifi NDefmessage succeeded !`
9. Now on your phone select “Read tags” and place the phone over the card.
10. You will see a section labeled “Content (tap to launch)” with the network information you just typed in. Tap that and you will see a window appear asking if you are sure you want to connect to that network. Tap “Yes”.
11. Your phone should now be connected to the network.



Pairing a Device to a WI-FI Network

Android Version Marshmallow 5.x.x

1. Do a Ctrl-C in the terminal to stop the program.
2. Type `"handOverDemo --help"` look over the flags. Note the double dashes and the single dashes in the command! Note the various authentications and encryptions supported.
3. I will provide the below settings delineated by "???".
4. Type in `"handOverDemo --wifi -ssid NXP-HOW -pwd nxp1234567 -auth WPA2PSK -enc AES"`
5. Place the phone over the Reader Antenna to write the data and connect to the network.
6. You should see `"Push Wifi NDefmessage succeeded !"`
7. You should see a window appear asking if you are sure you want to connect to that network. Tap "Yes".
8. Your phone should now be connected to the network.

Here is the code that built that message

```
bool LibNfcManager::writeNdefWifiMessage(int mTagHandle,unsigned char* pSsid,unsigned char* pPwd, unsigned char* pAuth, unsigned char* pEnc)
{
    unsigned char* lWifiMessage = (unsigned char*)malloc(sizeof(unsigned char)*255);
    int lWifiMessageLength,ret = 0;
    createWifiMessage(pSsid, pPwd, pAuth, pEnc, lWifiMessage, &lWifiMessageLength);
    ret = nfcTag_writeNdef(mTagHandle, lWifiMessage, lWifiMessageLength);
    free (lWifiMessage);
    if(ret == 0)
        return true;
    else
        return false;
}

void LibNfcManager::createWifiMessage(unsigned char* pSsid,unsigned char* pPwd,unsigned char* pAuth,unsigned char* pEnc,unsigned char* pMsg,int *pMsgLength)
{
    int currentIndex =0;

    unsigned char header = 0xD2;
    const char *type = "application/vnd.wfa.wsc";
    size_t typeLength = sizeof(unsigned char)*strlen((char*)type);

    size_t pwdLength = (unsigned char) (sizeof(unsigned char)*strlen((char*)pPwd));
    unsigned char pwdSettings[] = {0x10,0x27,0x00,(unsigned char)pwdLength};

    size_t ssidLength = (unsigned char) (sizeof(unsigned char)*strlen((char*)pSsid));
    unsigned char ssidSettings[] = {0x10,0x45,0x00,(unsigned char)ssidLength};

    unsigned char authSettings[] = {0x10,0x03,0x00,0x02,0x00};

    unsigned char encSettings[] = {0x10,0x0F,0x00,0x02,0x00};

    size_t wifiInfoLength = (unsigned char) ( pwdLength + sizeof(pwdSettings) + ssidLength + sizeof(ssidSettings) + 0x01 + sizeof(authSettings) + 0x01 + sizeof(encSettings));

    unsigned char wifiInfoSettings[] = {0x10,0x0E,0x00,(unsigned char)wifiInfoLength};
    size_t payloadLength = (unsigned char) (wifiInfoLength + sizeof(wifiInfoSettings));
    *pMsgLength = payloadLength + typeLength + 3;

    //build message
    pMsg[currentIndex] = header;
    currentIndex += 1;
    pMsg[currentIndex] = typeLength;
    currentIndex += 1;
    pMsg[currentIndex] = payloadLength;
    currentIndex += 1;
    memcpy(&pMsg[currentIndex],type,typeLength);
    currentIndex += typeLength;

    memcpy(&pMsg[currentIndex],wifiInfoSettings,4);
    currentIndex += 4;
    memcpy(&pMsg[currentIndex],ssidSettings,4);
    currentIndex += 4;
    memcpy(&pMsg[currentIndex],pSsid,ssidLength);
    currentIndex += ssidLength;
    memcpy(&pMsg[currentIndex],pwdSettings,4);
    currentIndex += 4;
    memcpy(&pMsg[currentIndex],pPwd,pwdLength);
    currentIndex += pwdLength;

    memcpy(&pMsg[currentIndex],authSettings,5);
    currentIndex += 5;
    memcpy(&pMsg[currentIndex],pAuth,1);
    currentIndex += 1;

    memcpy(&pMsg[currentIndex],encSettings,5);
    currentIndex += 5;
    memcpy(&pMsg[currentIndex],pEnc,1);
    currentIndex += 1;

    return;
}
```



Wi-Fi Simple Configuration Technical Specification

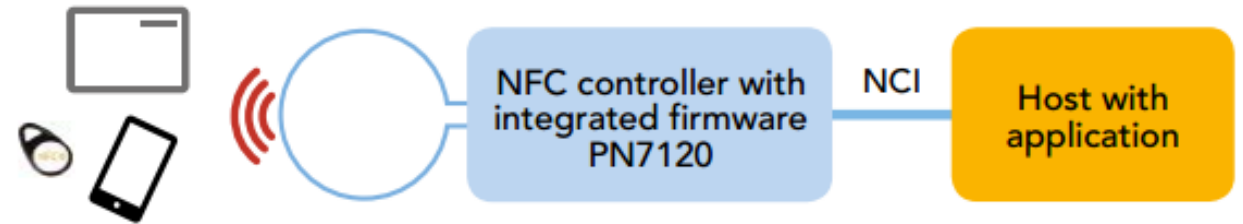
Version 2.0.3.21

SUMMARY AND SUPPORTING MATERIAL

PN7120

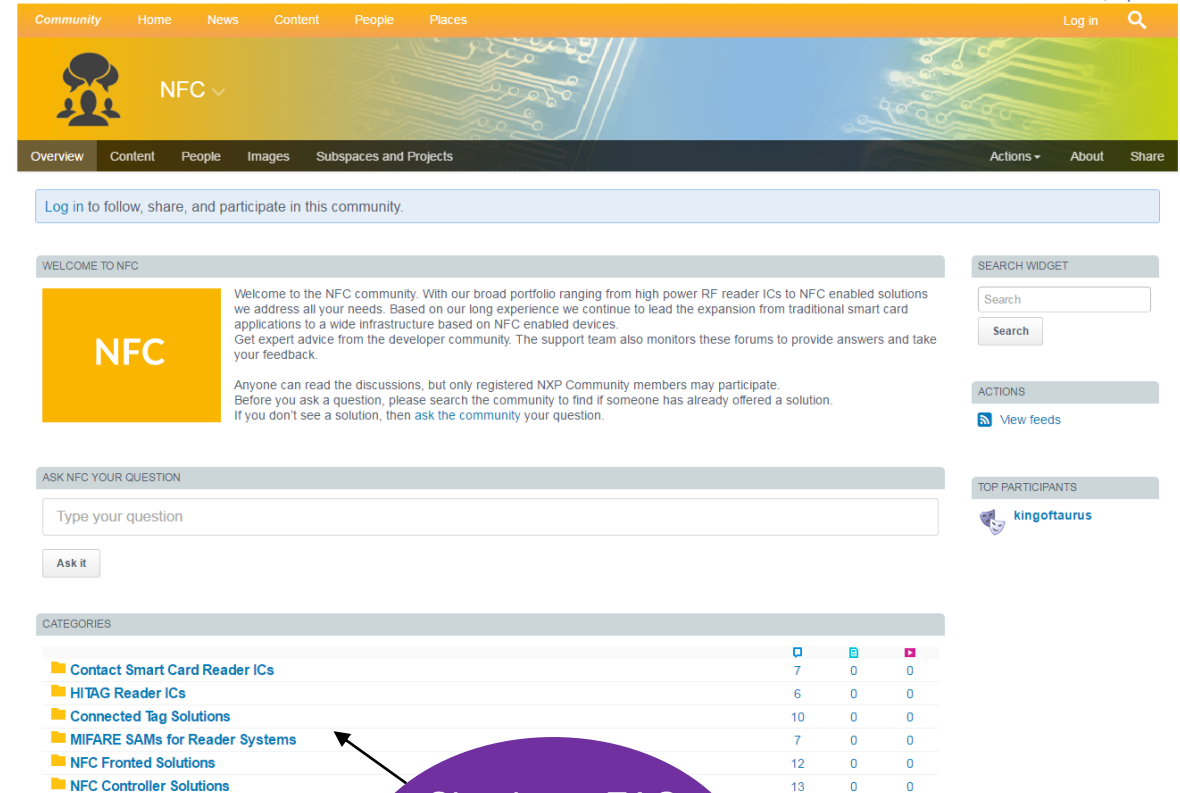
Key points

- Highly integrated and full NFC Forum controller solution
- Multiple target markets:
 - TVs, set-top boxes. Home automation, home appliances, wearable, printers, IP phones, gaming consoles, healthcare, wireless routers, etc.
- Multiple use cases:
 - Pairing, Personalization, User Interface, Maintenance, Logical access control, etc
- RF protocols:
 - R/W: ISO/IEC 14443 Type A/B, ISO/IEC 15693, FeliCa
 - P2P: ISO/IEC 18092 (Active and Passive)
 - CE: ISO/IEC 14443 Type A/B
- Ease of integration:
 - Pre-loaded FW, NCI interface, Linux and android drivers
- Low bill of materials :
 - Direct connection to application host, BGA package
- Low-power operation mode
- Demo kit and support information available



PN7120 and Contactless Technical Trainings and Additional Information

- [NFC Technology Hub](#)
- [PN7120 product support information](#)
- [NFC and Reader Ics](#)
- [Complete Products Webinars Archive](#)
- [PN7120 NFC in Linux - Get started with the PN7120S controller board](#)
- [Smart Home NFC commissioning solution](#)
- [NFC Forum](#)



The screenshot shows the NXP Community website interface. At the top is a navigation bar with links for Community, Home, News, Content, People, and Places. Below this is a header banner with the NXP logo and the text 'NFC'. A secondary navigation bar includes Overview, Content, People, Images, Subspaces and Projects, Actions, About, and Share. A blue box prompts users to 'Log in to follow, share, and participate in this community.' Below this is a 'WELCOME TO NFC' section with a welcome message and a call to action to ask a question. A search widget is on the right. The 'ASK NFC YOUR QUESTION' section has a text input field and an 'Ask it' button. The 'CATEGORIES' section lists various topics with corresponding counts.

CATEGORIES			
Contact Smart Card Reader ICs	7	0	0
HiTAG Reader ICs	6	0	0
Connected Tag Solutions	10	0	0
MIFARE SAMS for Reader Systems	7	0	0
NFC Fronted Solutions	12	0	0
NFC Controller Solutions	13	0	0

Check our FAQ and community [NXP Community](#) for latest posts on PN7120

Renke Bienert

- RF engineer since 1996
- RFID and NFC Expert since 2001
- Technical consulting for e.g.
 - RFID ticketing of the FIFA world cup 2006
 - the development of the Technical Guidelines for the Secure Use of RFID (BSI TR-03126)
 - standardization of contactless payment at EMVco
- Chairman of the DIN working group NIA 17.8 (until 2009)
- Member of various DIN and ISO/IEC working groups (until 2009)
- Co-Author of the book “RFID - MIFARE and Contactless Smartcards in Application“, Elektor International Media BV 2013, ISBN 978-1-907920-14-1



NFC TECHNOLOGY HUB

Your source for everything NFC

www.nxp.com/nfc

- Latest news
- Latest product news
- Technical NFC Community
- Downloads
- And more to discover...

NFC Technology Hub

Near Field Communication is hot. In today's increasingly connected world, this simple, intuitive technology lets you interact securely with the world around you with a simple touch. NFC is available in hundreds of millions of smartphones, tablets, and other consumer electronics, with new devices arriving almost daily. We are convinced to see NFC everywhere very soon. This hub gives you technology insights as well as the latest news about NFC solutions from NXP.

With NFC being a specialized subset of RFID, also check out our dedicated [RFID technology page](#).

NFC News

[NFC pairing - More time to relax, entertain, and connect at home](#)

With just a tap, new purchases can perform service discovery, connect to the home network, or pair with other components, such as high-end speakers...

[Blog: the future of mobile transit](#)

With NFC (PN66T) in your phone and wearable, you can securely preload your fare into the phone with an instant online purchase...

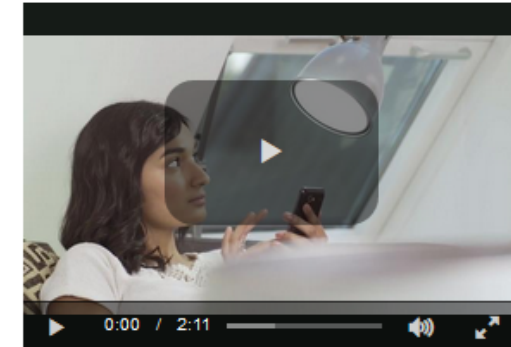
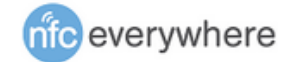
[Press Release: NXP and Xiaomi Announce Mobile Payment Partnership](#)

[Read more about NFC >](#)

NFC Products

When adding NFC to a system, there are three options to choose from: [NFC frontends](#), which provide just the NFC function, [NFC controllers](#), which combine the NFC frontend with a microcontroller, and [NFC connected tag ICs](#), which are passive microchips used in smart NFC tags. We have released new products in all categories ushering in a new era in the evolution of NFC to bring intuitive proximity technology everywhere:

- [PN5180](#): High-power NFC frontend solution
- [PN7462](#): NFC Cortex-M0 microcontroller offering high performance and low-power consumption
- [NTAG 124 plus](#): NFC Forum Type 2 Tag compliant IC with PC interface



NFC Commissioning for Smart Homes (02:11 min)

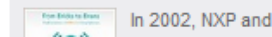
Design Resources

- [NFC Knowledge Base](#)
- [NFC Applications](#)
- [Documentation](#)
 - [NFC Everywhere: Controller, frontend, and connected-tag solutions for the next generation of NFC applications \(Brochure\)](#)
 - [NFC for embedded applications: Your critical link for the Internet of Things \(Brochure\)](#)
 - [Loader Service: The Tipping Point for Secure NFC Payments \(Whitepaper\)](#)
 - [What NFC means for smart factories, intelligent supply chains, and Industry 4.0 \(Whitepaper\)](#)
 - [NFC Product portfolio](#)

Featured Videos



History of NFC



NFC Support



THANKS FOR ATTENDING!





SECURE CONNECTIONS
FOR A SMARTER WORLD

ATTRIBUTION STATEMENT

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, CoolFlux, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE Classic, MIFARE DESFire, MIFARE Plus, MIFARE F1eX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TrenchMOS, UCODE, Freescale, the Freescale logo, AltiVec, C 5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, and UMEMS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. ARM, AMBA, ARM Powered, Artisan, Cortex, Jazelle, Keil, SecurCore, Thumb, TrustZone, and μ Vision are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. ARM7, ARM9, ARM11, big.LITTLE, CoreLink, CoreSight, DesignStart, Mali, mbed, NEON, POP, Sensinode, Socrates, ULINK and Versatile are trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org. © 2015–2016 NXP B.V.

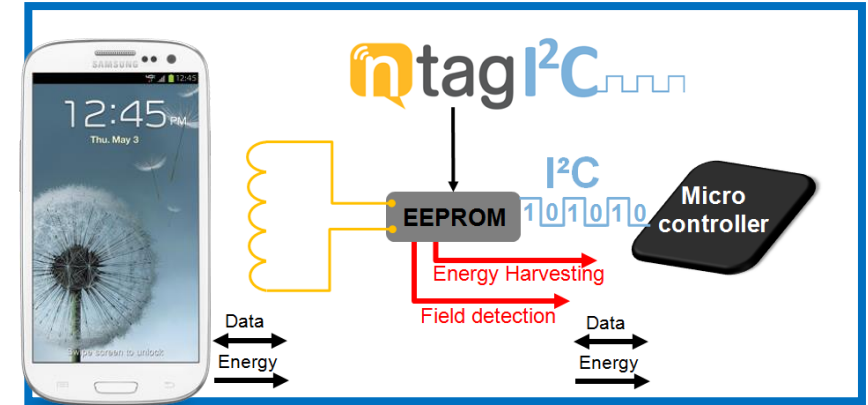


CONNECTED TAGS



NTAG I²C *Plus* Features

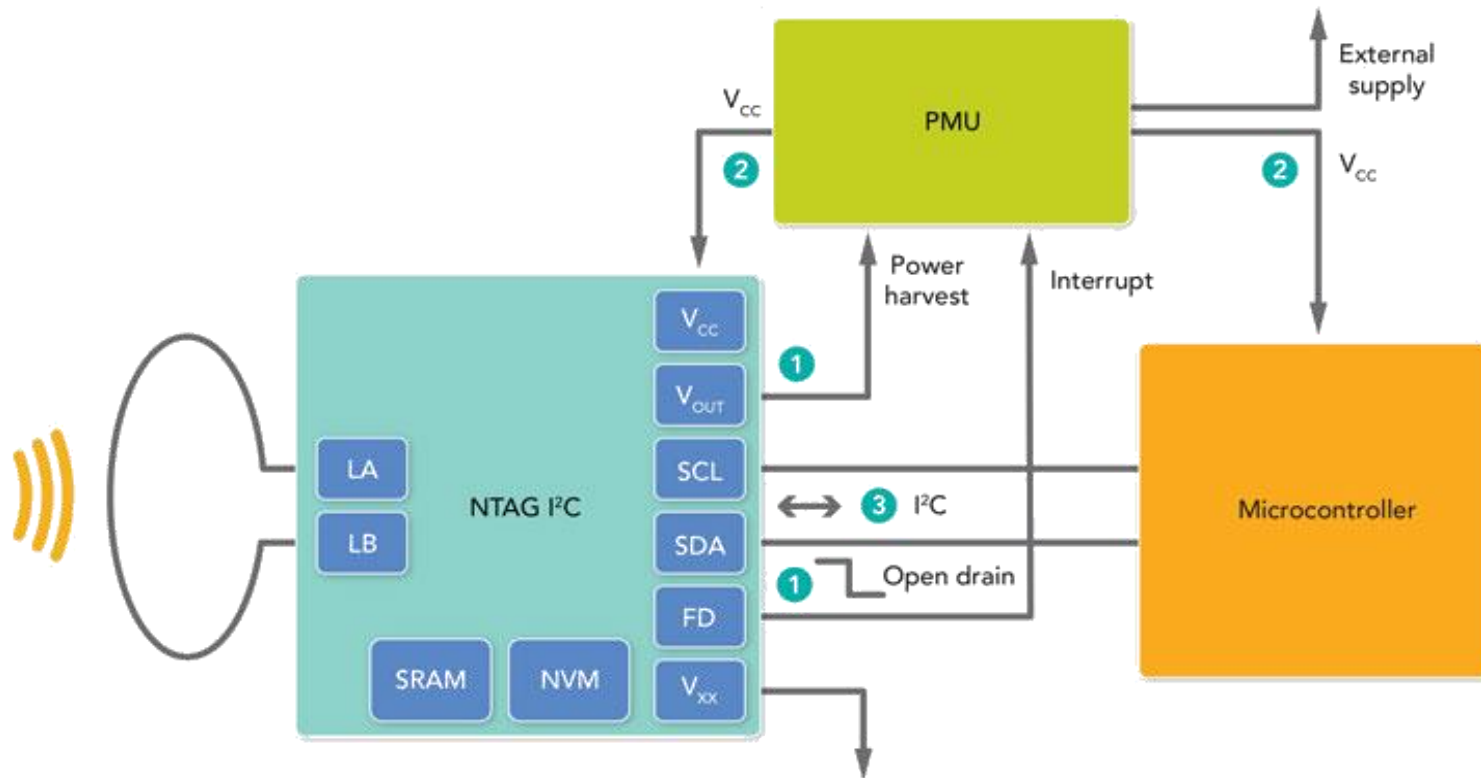
- NFC Forum Type 2 Tag (will be possible to certify)
- Dual interface
 - NFC & I²C (100 & 400 kHz) interface
 - **50 pF input capacitance**
- Either 888 or 1912 Byte EEPROM
- Energy harvesting functionality to power external device
- PIN to PIN outlines & fully backward compatible



What's NEW?

- **Pass Through Mode performance** increased by factor 4 in NFC to I²C with FAST_WRITE command to SRAM
- **Authentication:** Originality signature based on ECDSA (compatible with the NTAG 21x(F) family)
- **Access protection:** Optional 32-bit password on the full EEPROM and the SRAM buffer
- **SO8 package** & XQFN8 package (1.6 x 1.6 x 0.5 mm) & TSSOP8
- **Increased Compatibility** with legacy NFC devices (SRAM and CONFIG on sector 0)

NFC Commissioning Solution



- For power management, the whole system may be powered up upon Field Detection
- FD is asserted low when the field is detected; PMU receives interrupt, switches V_{CC} on NTAG and μ C; μ C activates I²C
- Alternatively, V_{OUT} (harvested energy) may be used for low power devices as source of power.