AN10990

Example Projects for NXP RD710/RD852 Readers

Rev. 1.0 — 1 April 2011
197910

Application note
COMPANY PUBLIC

Document information

Info	Content
Keywords	Example Projects Pegoda, Pegoda, MIFARE examples, Pegoda MIFARE Plus, Pegoda MIFARE Ultralight, Pegoda, MIFARE Ultralight C, Pegoda SAM modes
Abstract	This document provides illustrative information on setting up the Pegoda software stack and specific mode configurations.

Revision history

Rev	Date	Description
1.0	20110401	First release.
	20101101	Draft version

Contact information

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

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

1. Introduction

1.1 Scope

The Pegoda development kit is delivered with an Example Project code solution. The goal of this project is to offer development people everything they need for rapid development.

The main target of this document is focused on explaining the steps to build up the software stack for different reader modes. For correct reader and chip configuration, different layer dependent objects must be generated and linked.

Protocol related communication to MIFARE cards is not scope of this document. For more information on card commands and how they are used, refer to the Example Project source code, the “NXP Reader library” document and the MIFARE application notes.

1.2 Audience

This document is intended for use by manufacturers wanting to develop applications based on the software stack delivered within the Pegoda package.

1.3 Applicable Documents or References

[1] [ISO/IEC 14443]

[2] PC/SC Workgroup Specifications www.nxp.com/redirect/pcscworkgroup.com/specifications/specdownload

[3] NXP Reader Library (in .chm format on CD included)

[4] www.nxp.com/redirect/en.wikipedia.org/wiki/Interrupt_handler

1.4 Acronyms and Abbreviations

SAM	Secure Access Module
S	SAM in S-Mode
N	no SAM
X	SAM in X-Mode
BAL	Bus Abstraction Layer
HAL	Hardware Abstraction Layer
PAL	Protocol Abstraction Layer
AL	Application Layer

1.5 Requirements

Opening the Example Project Solution requires Microsoft Visual Studio 2005 or later.

Furthermore, a Pegoda reader is necessary to execute the examples.

For information on availability of samples as well as documentation, please refer to the application note ‘Pegoda EV710/EV852 Documentation and Sampling guide’.

2. Overview

The Example Project provides developers with plenty of card communication examples based on MIFARE card products to start rapid development of applications. Examples included are

- MIFARE Classic (Authentication, Read/Write Block)
- MIFARE Ultralight, MIFARE Ultralight C
- MIFARE Plus (Security Level 3)

Programming applications with the Pegoda kit requires a basic knowledge of the different reader modes and the BFL structure.

Note, the complete command set is described in the NXP Reader library [3]. In order to use the command set within a new application project, the NXPRdLib.dll has to be included in the project file.

Building the software stack for a specific reader mode requires different software models to be generated.

The three types of operating modes within the PC/SC mode are:

- **No SAM mode (only RD710)**

  The most important aspect of this mode is performing activation and polling sequence for ISO 14443 A type cards as defined in PC/SC part 3. Selected cards are put in slot manager and notification is sent to PC/SC driver. There can be only one ISO 14443-3 card or multiple (maximum 14) ISO 14443-4 cards in the field.

- **SAM in non X-mode**

  SAM in non X-mode is similar as No SAM mode. The only difference is that slot 0 is always occupied with SAM, which can be used as key store, cryptographic engine, etc. The reader chip cannot be controlled by SAM in this mode.

- **SAM in X-mode**

  In this mode communication is only through SAM (slot 0). Only limited number of proprietary commands can be executed.

The specific (SAM) mode has to be specified during initialization of the software components.

2.1 NXP Reader Library Overview

The NXP reader library is encapsulated into Layers and Components written in ANSI C. The library structure provides a modular way of programming and setting up the reader interface.

The reader library consists of 4 layers

- BAL (Bus Abstraction Layer)
- HAL (Hardware Abstraction Layer)
- PAL (Protocol Abstraction Layer)

- AL (Application Layer)

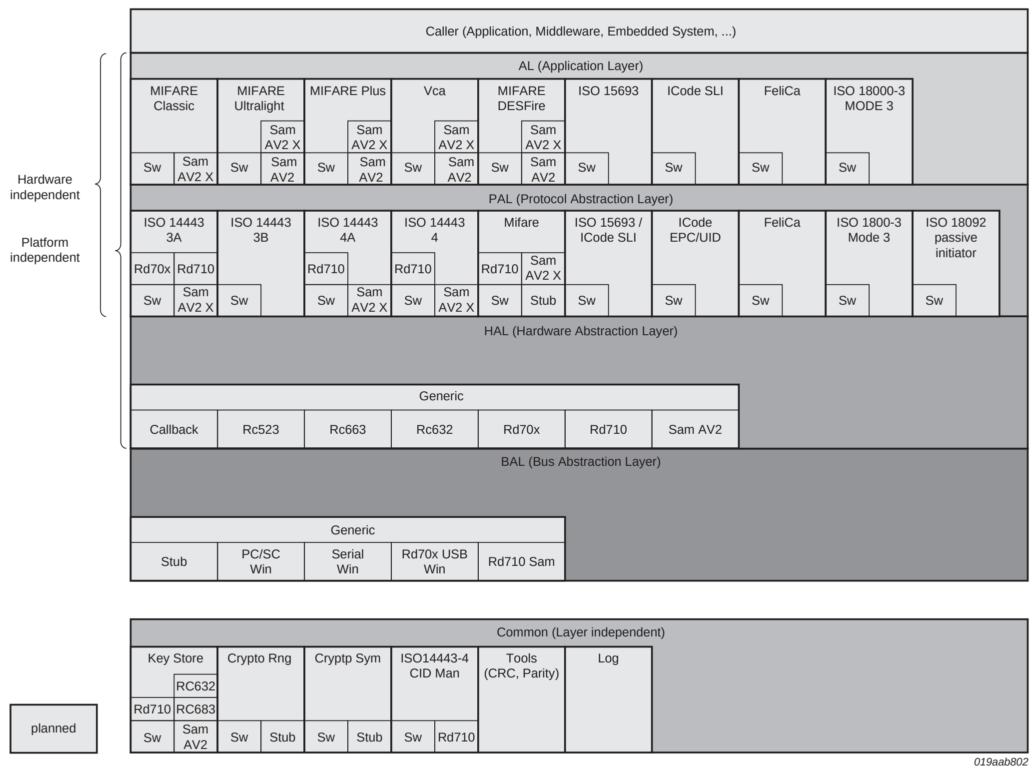

Fig 1. Reader Library Model

Each Layer consists of different components having a generic interface and a specific implementation.

2.2 Building the stack

In order to use the software library a stack of components has to be build up from bottom (BAL) to top (AL) layer. Fig 2 shows the various elements to build up a full software stack on the PC site for contactless card communication.

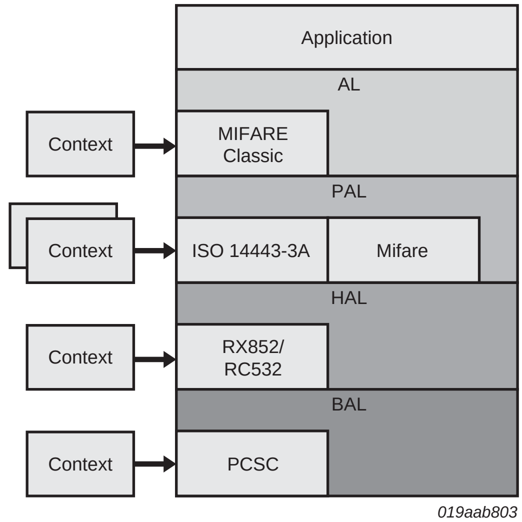

Fig 2. Building up the stack

Every component has to be initialized before usage. E.g. Initialization of the BAL layer requires specific context or data parameter to be fed into the component.

BAL Init

Therefore, Rd710 USB (Windows) BAL parameter structure has to be initialized

```
phbalReg_PcscWin_DataParams_t balPcsc;
```

and passed to the init function of the component

```
status = phbalReg_PcscWin_Init(&balPcsc, sizeof(phbalReg_PcscWin_DataParams_t));
```

Every component has an Init function for context data and component initialization. The init function checks the context data length to ensure no buffer overflow.

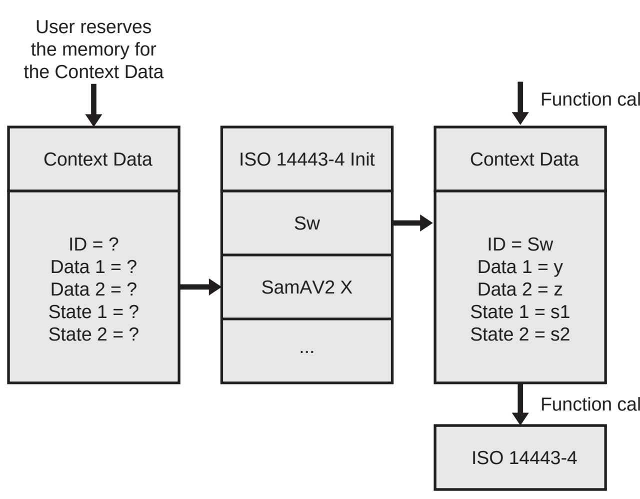

Fig 3. Context handling

The correct reader port parameter must be set with the setPort-function. Depending on the reader mode, different parameters are possible.

```
status = phbalReg_SetPort(&balReader, /**< [In] Port Name as String. */);
status = phbalReg_OpenPort(&balReader);
```

The next layer can now be built up using the same procedure.

HAL init

The HAL layer requires the structure

```
phhalHw_Rd710_DataParams_t halRd710;
```

to be initialized first.

The init routine is called by the HAL-structure parameter and referenced to the bottom BAL layer:

```
status = phhalHw_Rd710_Init(
          &halRd710,
          sizeof(phhalHw_Rd710_DataParams_t),
          &balPcsc,
          bHalBufferReaderTx,
          sizeof(bHalBufferReaderTx),
          bHalBufferReaderRx,
          sizeof(bHalBufferReaderRx));
```

Depending on the connected reader type (SAM, no SAM, …) it might be necessary to initialize additional structures. Refer to the examples in the code for more information.

PAL Init

The protocol abstraction layer inherits implementation of card activation and card protocol. Dependent on the card to be operated in the field, specific objects have to be initialized. For communication with a MIFARE Ultralight card on a Pegoda in non-SAM mode for example, we have to call the following structures:

```
status = phpalI14443p3a_Sw_Init(&palI14443p3a, sizeof(palI14443p3a), pHal);
CHECK_SUCCESS(status);

status = phpalI14443p4a_Sw_Init(&palI14443p4a, sizeof(palI14443p4a), pHal);
CHECK_SUCCESS(status);

status = phpalI14443p4_Sw_Init(&palI14443p4, sizeof(palI14443p4), pHal);
CHECK_SUCCESS(status);

status = phpalMifare_Sw_Init(&palMifare, sizeof(palMifare), pHal,&palI14443p4);
CHECK_SUCCESS(status);
```


Then, the HAL has to be configured for type-A cards with command

```
status = phhalHw_ApplyProtocolSettings(pHal, PHHAL_HW_CARDTYPE_ISO14443A);
```

Different card settings are defined in the header file ***phhalHw.h.***

AL operations

The application layer is the top layer of the software stack, providing specific implementations of various contactless protocols. The activation of the card must be done upfront in the lower protocol abstraction layer.

For detailed examples refer to the source code in the Example project.

Layer independent components

The software keystore and cryptographic functions in the library are not part of the layered approach. Before using any crypto operations within the provided functions, the dedicated crypto implementation has to be initialized:

```
phStatus_t phCryptoSym_Sw_Init  ( phCryptoSym_Sw_DataParams_t *  pDataParams,
  uint16_t  wSizeOfDataParams,
  void *  pKeyStoreDataParams
 )
```

3. How to set up different reader modes

The following subchapters should provide an entry point for understanding the basic handling of different reader modes and setting up the software stack, respectively.

Some points to remember from chapter 2:

- In order to use the library, the NXPRdLib software stack has to be built up from bottom to top layer
- Every component has dependencies to a component on the same layer or below
- For every card a complete stack has to be built to be able to communicate with the card
- The HAL layer provides the functionality to switch between multiple cards.

3.1 Single ISO14443 card activation

The basic example of activation a single card and initializing the components and layers are already described in chapter 2.2. The stack model is outlined again in Fig 4.

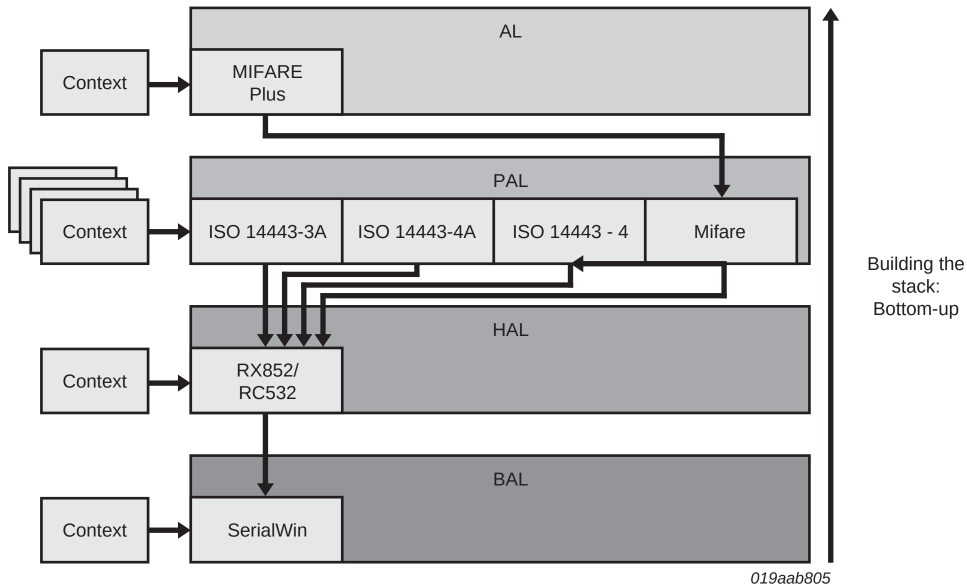

Fig 4. Single ISO14443 card activation

3.2 Multiple ISO14443 card activation

Communicating to more than one card requires initialization of additional HAL layers on a common BAL layer:

```
phhalHw_Rd710_DataParams_t halRd710[1];
phhalHw_Rd710_DataParams_t halRd710[2];
phhalHw_Rd710_DataParams_t halRd710[x];
```

All *halRD710[x]* objects have to be passed to the Init function to specify the BAL path.

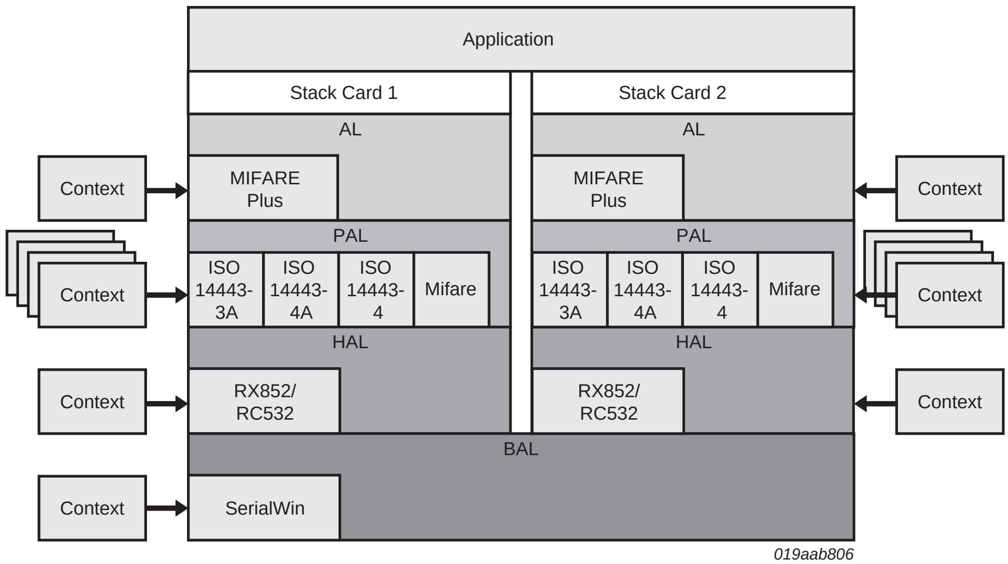

Fig 5. Multiple ISO14443 card activation

3.3 SAM in X-Mode single card activation

The configuration of the Pegoda with SAM in X-Mode is different to non-SAM mode in the setup of the stack. Among others, it requires the ISO14443 stack to be built-up on the SAM hardware. More details will follow; let's see the steps to be done first:

1. The BAL PCSC components have to be initialized
   a. `phbalReg_PcscWin_DataParams_t balPcsc;`
   b. `phbalReg_PcscWin_Init(&balPcsc, sizeof(phbalReg_PcscWin_DataParams_t));`
2. Set the port to SAM in X-mode
   a. `phbalReg_SetPort(&balPcsc, (uint8_t*)PCSC_READER_P2_X_NAME);`
3. Generate an object for the SAM-HAL structure and initialize with the SamAVs_Init function
   a. `phhalHw_SamAV2_DataParams_t halSamLC0;`
   b. `phhalHw_SamAV2_Init(&halSamLC0,…)`
4. A keystore object can be created and a separate logical channel must be linked to this keystore object.
   a. `phKeyStore_SamAV2_DataParams_t keyStoreSAM;`
   b. `phhalHw_SamAV2_DataParams_t halSamLC1;`
   c. `phhalHw_SamAV2_Init(&halSamLC1,…)`
   d. `phKeyStore_SamAV2_Init(&keyStoreSAM, sizeof(phKeyStore_SamAV2_DataParams_t), &halSamLC1);`

Now, two logical channels have been created, one for the communication to the card, the other is linked to the keystore on the SAM

5. The reader can be opened now with
   ```
   a.  phbalReg_OpenPort(&balPcsc);
   ```
6. So far the usage and initialization of the SAM structures were transparent. This means, we didn't differentiate between SAM AV1 or AV2 or even the connected Pegoda reader. We just generated SamAV2_DataParams objects for the keystore and the communication channel. Now it's time to get the SAM version with the getconfig command
   ```
   a.  phhalHw_GetConfig(&halSamLC0, PHHAL_HW_SAMAV2_CONFIG_HOSTMODE,
       aVersion);
   ```
   This is necessary in order to load the DES or AES key into the corresponding SAM version and format the key entry correctly:
   ```
   b.  phKeyStore_FormatKeyEntry(pKeyStore, SAM_Key, /*Key_Type*/);
   ```
7. Before we can communicate with the card, the ISO14443-3 and ISO14443-4 PAL layers have to be generated on the SAM. Therefore, we have to call SAM specific objects:
   ```
   a.  phpalI14443p3a_SamAV2_X_DataParams_t I14443p3a_X; /**< SAM PAL-
       ISO14443P3A parameter structure */
   b.  phpalI14443p4a_SamAV2_X_DataParams_t I14443p4a_X; /**< SAM PAL-
       ISO14443P4A parameter structure */
   c.  phpalI14443p4_SamAV2_X_DataParams_t I14443p4_X;   /**< SAM PAL-
       ISO14443P4 parameter structure */
   ```
   This is different to the software based approach on the non SAM reader mode. Whenever it is necessary to generate the PAL stack on the PC/software side, we need to generate objects with
   ```
   d.  phpalI14443p3a_Sw_DataParams_t I14443p3a_Sw;
   e.  …
   ```
8. As usual the pointer of the generated objects and the underlying HAL layer must be linked with the corresponding Init functions:
   ```
   a.  phpalI14443p4a_SamAV2_X_Init((&I14443p4a_X,
       sizeof(phpalI14443p4a_SamAV2_X_DataParams_t), pHal);
   b.  phpalI14443p4_SamAV2_X_Init(&I14443p4_X,
       sizeof(phpalI14443p4_SamAV2_X_DataParams_t), pHal);
   c.  phpalMifare_SamAV2_X_Init(&palMifare_X,
       sizeof(phpalMifare_SamAV2_X_DataParams_t), pHal, &I14443p4_X);
   d.  phalMfc_SamAV2_X_Init(&alMfc_X, sizeof(phalMfp_SamAV2_X_DataParams_t),
       pHal, &palMifare_X);
   ```
9. Now we can perform host authentication and can communicate with the card

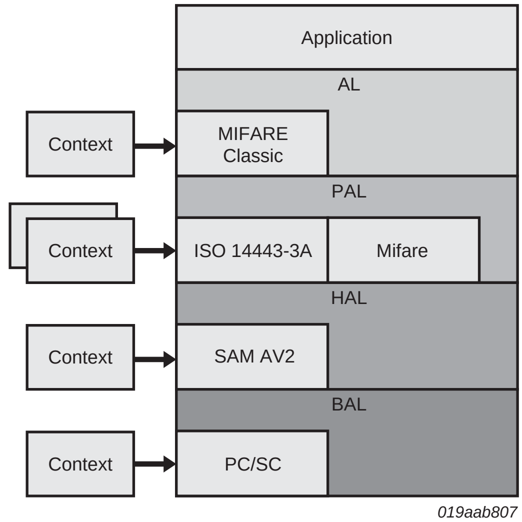

Fig 6. SAM in X-Mode

3.4 SAM in non X-Mode single card activation

1. The BAL PCSC components have to be initialized
   a. `phbalReg_PcscWin_DataParams_t balPcsc;`
   b. `phbalReg_PcscWin_Init(&balPcsc, sizeof(phbalReg_PcscWin_DataParams_t));`
2. Set the port to SAM in **non** X-mode
   a. `phbalReg_SetPort(&balPcsc, (uint8_t*) PCSC_READER_P2_NON_X_NAME);`
3. A HAL RD710 component has to be initilized
4. A second stack based on the RD710SAM BAL has to beinitialized and linked to the PCSC BAL as outlined in Fig 7.
   a. `phbalReg_Rd710Sam_Init(&balRd710Sam, sizeof(phbalReg_Rd710Sam_DataParams_t), &balPcsc, aBalBufferTx, sizeof(aBalBufferTx), aBalBufferRx, sizeof(aBalBufferRx));`
5. The last step would be the initiaization of the SAM logical channel component and opening the PCSC port.

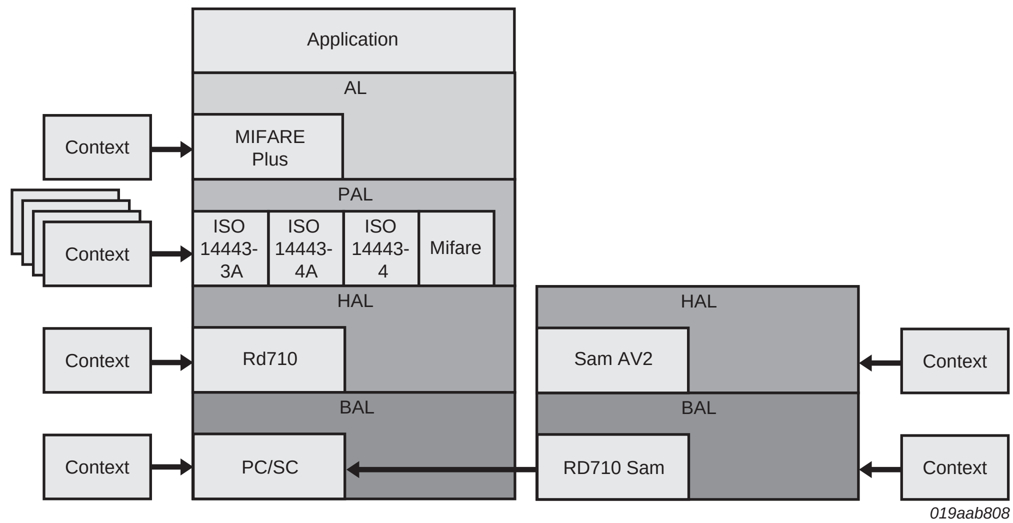

Fig 7. SAM in non X-Mode

4. Legal information

4.1 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.

4.2 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 national 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.

4.3 Licenses

Purchase of NXP ICs with ISO/IEC 14443 type B functionality

This NXP Semiconductors IC is ISO/IEC 14443 Type B software enabled and is licensed under Innovatron's Contactless Card patents license for ISO/IEC 14443 B.

The license includes the right to use the IC in systems and/or end-user equipment.

RATP/Innovatron Technology

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

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

5. List of figures

6. Contents

Date of release: 1 April 2011
197910

Document identifier: AN10990