AN10927

MIFARE and handling of UIDs

Rev. 3.1 — 02 October 2013
190731

Application note
COMPANY PUBLIC

Document information

Info	Content
Keywords	Single Size UID, Double Size UID, 4 Byte UID, 7 Byte UID, SNR, NUID, FNUID, ONUID
Abstract	This document shows the use of UIDs in contactless smartcard systems. It indicates recommendations about the Random ID, mixed use of 4 byte and 7 byte UIDs in the same system, and it describes the options how to upgrade 4 byte UID systems to accept 7 byte UID smart cards.

Revision history

Rev	Date	Description
3.1	20131002	Typos corrected in Cascade Level 3 (Section 2) Ultralight EV1 and MIFARE4Mobile added in Table 3 (Section 2.4) UID with shortcut for MIFARE from mobile added (Section 3.1.2) CLRC663, CRC630, CLRC631, PR601, PRH601 added in Annex A Annex C (Source code to derive NUID out of a Double Size UID) added
3.0	20110804	MIFARE Classic next generation added.
2.0	20100901	Bit order corrected (Section 3.2.2), 7 byte MF1 ICS x0 added in Table 4 (Section 3.2.5), Table 3 updated (Section 2.4)
1.0	20100519	Initial version

Contact information

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

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

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

© NXP B.V. 2013. All rights reserved.

1. Introduction

This document shows the use of UIDs in contactless smartcard systems. It indicates recommendations about the use of Random ID, the mixed use of 4 byte (single size) and 7 byte (double size) UIDs in the same system, and it describes the options how to upgrade 4 byte UID systems to use 7 byte UID smart cards.

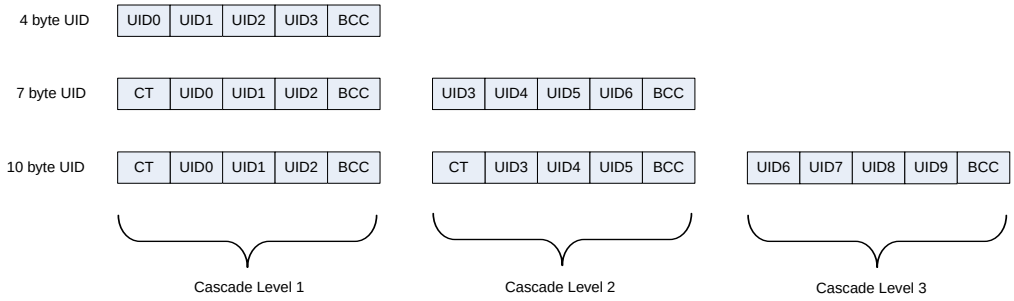
Note: A UID is not a “serial number”, but a unique identifier. There is no recommendation how to turn the array of bytes into an integer.

Note: “UID” is a common expression, defined in the ISO/IEC 14443-3. In some case the UID is even not unique (like RID or NUID, see below).

Note: The 4 byte UID is called “Single Size UID”, too. The 7 byte UID is called “Double Size UID”, too. The 10 byte UID is called “Triple Size UID”, too.

2. MIFARE and ISO/IEC 14443 UIDs

In this section the use of UIDs according to the ISO/IEC 14443 is described. Fig 1 shows the three different UID sizes defined in ISO/IEC 14443-3 as they are used during the anti-collision and selection procedure.



(1) BCC = Block Check Character, it is calculated as exclusive-or over the 4 previous bytes.

(2) CT = Cascade Tag, to indicate a following cascade level.

Fig 1. UIDs according to ISO/IEC 14443

Fig 2 shows the Anticollision sequence, which is a mandatory part of the card activation sequence. It automatically selects a single PICC with 4 byte UID (= Single Size UID), 7 byte UID (= Double Size UID) or 10 byte UID (= Triple Size UID).

Cascade Level 1

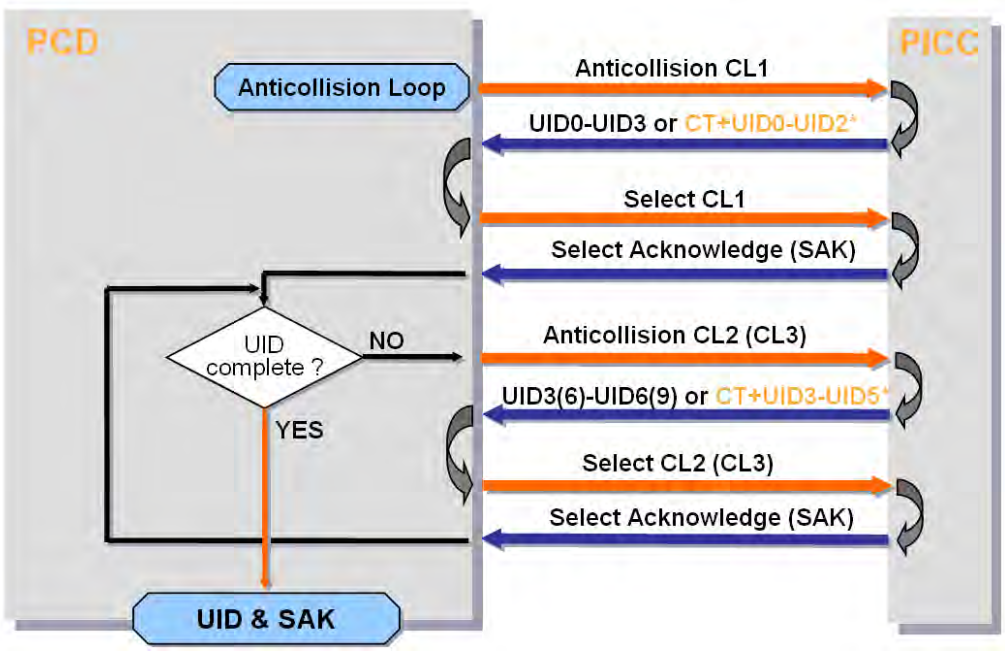
In the Cascade Level 1 the PCD sends the Anticollision command CL1 (0x93) and the PICC returns

- either the 4 byte UID (UID0...UID4) and one byte BCC,
- or a Cascade Tag (CT) followed by the first 3 byte of the UID (UID0...UID2) and one byte BCC.

The CT (0x88) indicates that the UID is not yet complete, and another Cascade Level has to follow.

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

© NXP B.V. 2013. All rights reserved.



(3) CT = Cascade Tag

(4) CL = Cascade Level

Fig 2. Anticollision sequence

Note: The UID0 byte of a 4 byte UID must not be 0x88.

The CL1 then must be selected, using the Select command CL1 (0x93). The PICC returns its SAK CL1, which indicates

- whether the UID is complete or not, and (if so),
  - the type of card (for details refer to [1] and [2]), and
  - whether the card supports T=CL.

Cascade Level 2

If the UID is not yet complete, the PCD continues with an Anticollision CL2 command (0x95), and the PICC returns

- either the last 4 bytes of the Double Size UID (UID3...UID6) and one byte BCC,
- or a Cascade Tag (CT) followed by the next 3 bytes of the Triple Size UID (UID3...UID5) and one byte BCC.

The CT (0x88) indicates that the UID is not yet complete, and another Cascade Level has to follow.

Note: The UID3 byte of a 7 byte or 10 byte UID must not be 0x88.

The CL2 then must be selected, using the Select command CL2 (0x95). The PICC returns its SAK CL2, which indicates

- whether the UID is complete or not, and (if so),
  - the type of card (refer to [1] and [2]), and

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

© NXP B.V. 2013. All rights reserved.

– whether the card supports T=CL.

Cascade Level 3

If the UID is not yet complete, the PCD continues with an Anticollision CL3 command (0x97), and the PICC returns

- the last 4 bytes of the Triple Size UID (UID6...UID9) and one byte BCC.

The CL3 then must be selected, using the Select command CL3 (0x97). The PICC returns its SAK CL3, which indicates

- the type of card (refer to [1] and [2]), and
- whether the card supports T=CL.

2.1 Single Size UID

The single size UID contains 4 bytes. As shown in Table 1, the value of the UID0 byte defines how those 4 bytes shall be interpreted.

Table 1. Assignment of Single Size UIDs

POR = Power on reset

UID0 [Hex]	Definition	Range
08	RID: UID1, UID2 and UID3 are dynamically generated during or after each Power-On-Reset (POR).	appr. 16 million
x0... x7	Proprietary use (i.e. used for MIFARE)	appr. 2.1 billion
18…78, 98…E8	Proprietary use (i.e. used for MIFARE)	appr. 218 million
x9...xE	Proprietary use (i.e. used for MIFARE)	appr. 1.6 billion
xF	Fixed number, non-unique	appr. 268 million
88	Cascade Tag	-
F8	RFU	-

Note: Single Size UIDs do not have a manufacturer code.

Note: The use of Single Size UIDs (unique ones) might end soon, since the number of usable IDs is limited to approximately 3.7 billion pieces only.

2.1.1 Random ID (RID)

A single size UID with UID0 = 0x08 indicates a Random Identifier. The Random ID (RID) is dynamically generated, when the PICC powers up. Deselecting a PICC does not reset the RID, but a field reset does.

Note: RID is always limited to 4 bytes.

Note: Depending on the PICC implementation, a UID (i.e. Double Size UID) may be retrieved from the card by proprietary means after the PICC is selected with its RID.

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

© NXP B.V. 2013. All rights reserved.

2.1.2 Fixed but non-unique ID (FNUID)

The 4 byte UIDs with UID0 = xF_h are fixed identifiers (like unique ones), but the same UID might be used for several PICCs, so that contactless systems cannot rely on the uniqueness of such a PICC identifier. These UIDs are called FNUID in the following.

The probability to have 2 PICCs on one PCD at the same time with the same FNUID is still extremely low.

However, it might create conflicts, if the contactless system uses the UID not only for the card activation but also as a logical reference to the PICC. There is a proposal how to handle this in chapter 3.2.

2.1.3 Re-used UID (ONUID)

The very old Single Size UIDs will be re-used, which means the same UID might be used for several PICCs, so that contactless systems cannot rely on the uniqueness of such a PICC identifier. These ID are called ONUID in the following.

The probability to have 2 PICCs on one PCD at the same time with the same ONUID is still extremely low.

However, it might create conflicts, if the contactless system uses the UID not only for the card activation but also as a logical reference to the PICC. There is a proposal how to handle this in chapter 3.2.

2.2 Double Size UID

Double Size UIDs always contain a manufacturer code in the UID0. With the double size UIDs each manufacturer can theoretically use up to $2.8 * 10^{14}$ UIDs.

2.2.1 Manufacturer Code

In double and triple size UIDs the UID0 contains the manufacturer code which indicates the manufacturer of the PICC as shown in Table 2.

Table 2. Manufacturer Code

UID0 [Hex]	Definition
81 ... FE	not allowed
04	NXP Semiconductors, formerly Philips Semiconductors

2.2.2 Unique ID ranges for Double Size UIDs

Double Size UIDs always contain a manufacturer code in the UID0.

Note: Due to the content of Double Size UIDs of MIFARE products the best diversification can typically be found in the UID1 and UID2.

2.3 Triple Size UID

Triple Size UIDs always contain a manufacturer code in the UID0.

Currently there is no PICC using a triple size UID. However, according to ISO/IEC 14443 it is mandatory that every PCD supports Triple size UIDs.

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

© NXP B.V. 2013. All rights reserved.

2.4 UID used in MIFARE products

In the past MIFARE Classic cards were limited to 4 byte UIDs only, i.e. normally every MIFARE Classic related product has used a single size UID only. Due to the limited number of UIDs in the single size range all new MIFARE related products are supporting 7 byte UIDs.

Table 3 indicates which MIFARE product uses which UID.

Table 3. UIDs and MIFARE products

NUID = Non Unique ID, ONUID = Re-used UID, FNUID = Fixed, non-unique UID

Product	MIFARE Ultralight™ (C)	MIFARE™ Classic		MIFARE Plus™	MIFARE DESFire™ (EV1)	SmartMX™	MIFARE4Mobile
	MF0 ICxx	MF1Syyyy	MF1SyyyyX[1]	MF1 PLUS	MF3 IC Dxx	P5 xx P6 xx	
Name	MIFARE Ultralight, Ultralight C Ultralight EV1	MIFARE Classic 1K/4K	MIFARE Classic 1K/4K	MIFARE Plus S, MIFARE Plus X (2K and 4K)	MIFARE DESFire, MIFARE DESFire EV1	with MIFARE implementation	
Single Size UID	-	-	-	-	-	x[2]	Depends which card is emulated.
Single Size FNUID	-	-	-	-	-	x	
Single Size ONUID	-	x	x	x	-	-	
Double Size UID	x	-	x	x	x	x[3]	
RID option	-	-	x	x[4]	x	x[3]	
UID Perso Option	-	-	X	-	-	x[3]	
UID needed for operation	-	x	x	x[5]	-	x[6]	
UID recommended for key diversification	x	x	x	x	x	x	

The Single Size FNUID or ONUID can be used like a Single Size UID – except the fact that identifier of this range will be used multiple times.

RID is optional and should be used to protect privacy. In case RID is enabled, there is a defined and confidential way to retrieve the UID for each product.

1. MIFARE Classic next generation
2. For existing masks using Singe Size UID only that have not been switched to Single Size FNUID yet
3. For MIFARE Classic implementation using the MIFARE FleX™ framework
4. MIFARE Plus support RID only in SL3.
5. In SL1 and SL2 only.
6. For the MIFARE Classic implementation.

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

© NXP B.V. 2013. All rights reserved.

3. UID and MIFARE Classic / MIFARE Plus

3.1 Card activation

In the past the MIFARE Classic always used a Single Size UID, some very old MIFARE readers may not have implemented the additional cascade levels according to the ISO/IEC 14443, which are required to select a Double Size UID. In such case there are the following different options to activate a card:

- Single Size NUID (FNUID or ONUID)
- RID

In any case it is strongly recommended to implement the full 4 byte, 7 byte and 10 byte UID card activation on the PCD, as required by the ISO/IEC 14443.

3.1.1 Single Size NUID

The MIFARE Plus card or MIFARE Classic card with Single Size NUID can be activated like a usual Single Size UID card.

Note: There is a very small probability that 2 cards in the PCD field have the same NUID, and therefore cannot be properly selected without the user removing one card.

Note: NUID might be an order option or an option which can be chosen during personalization of the card.

3.1.2 Double Size UID with “shortcut”

The MIFARE Classic next generation offers the feature to use the Double Size UID, but activate the card with REQA - Anticollision CL1 - Select CL1 – Read Block 0.

In such case the Read Block 0 command might return CRC and parity errors, if more than one card is selected. This conflict cannot be resolved by the reader, if it does not support CL2, but the user needs to separate a single card.

Note: The 4 bytes of the CL1 (CT + UID0...UID2) is taken as input for the MIFARE Classic authentication, if the MIFARE Classic next generation is selected with the Read Block 0.

Note: This feature is not supported for MIFARE Classic implementations using an NFC device for ISO/IEC 14443 protocol handling. As the “shortcut” functionality is not specified in ETSI/SCP TS 102 613, this feature is not supported. This needs to be taken into account when designing a contactless system which shall also support NFC devices.

Note: This feature is neither supported by the MIFARE Classic (MF1Syyyy) nor by the MIFARE Plus. Future versions of MIFARE Plus may include this feature.

Note: This feature is supported by the MIFARE Ultralight and MIFARE Ultralight C, too.

3.1.3 RID

Some MIFARE Classic, the MIFARE DESFire (EV1) and the MIFARE Plus offer the option to enable RID. RID is always 4 bytes only. The MIFARE Plus offers RID only in SL3.

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

© NXP B.V. 2013. All rights reserved.

3.2 UID in the contactless system

In some cases the reader infrastructure might be able to handle Double Size UIDs, but the (background) system can only handle 4 byte UIDs. Or vice versa, the reader infrastructure might not be able to handle Double Size UIDs, but the (background) system needs uniqueness and can handle Double Size UIDs.

In such a case there are at least 5 different options:

- Single Size NUID for card activation and for the system
- Single Size NUID for card activation, and Double Size UID for the system
- Double Size UID for card activation, and Single Size NUID for the system
- RID for card activation, and Single Size NUID for the system
- RID for card activation, and Double Size UID for the system

3.2.1 Single Size NUID for card activation and for the system

The MIFARE Plus card or MIFARE Classic card with Single Size NUID can be activated like a usual Single Size UID card.

Note: There is an extremely small probability that 2 cards in the field have the same NUID, and therefore cannot be properly selected without the user removing one card.

Note: NUID might be an order option or an option which can be chosen during personalization of the card.

There is a probability that the same NUID appears in the system more than once. Either the cards have to be pre-selected e.g. at issuing to avoid such collision in the system, or the system has to be able to deal with these cards in a special way.

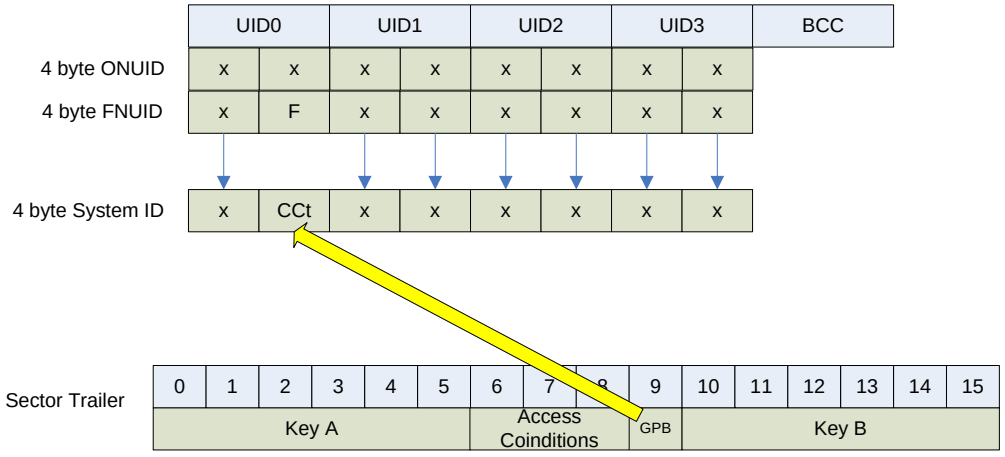
System ID

The system could use a 4 byte **system ID** (see Fig 3), derived from the

- high nibble of the UID0 (4 bit)
- the low nibble of the GPB used as card counter (4 bit)
- the UID1, UID2 and UID3.

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

© NXP B.V. 2013. All rights reserved.



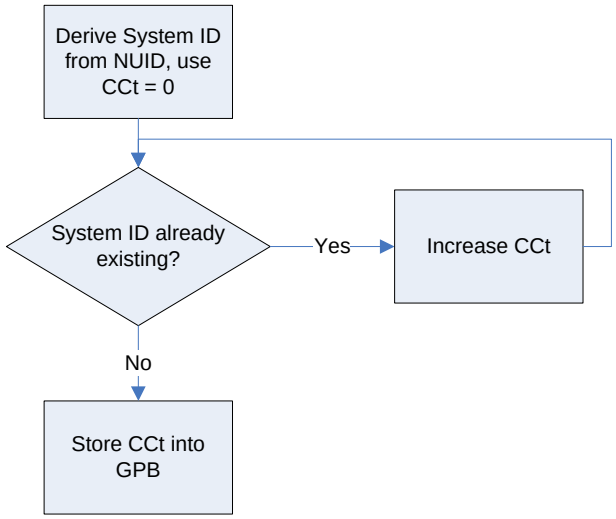
(1) CCt = CardCounter, to be stored in the GPB of an application sector

(2) GPB = General Purpose Byte (Byte 9 of a Sector Trailer)

Fig 3. How to derive a 4 byte System ID out of a NUID and GPB

This system ID must be created when the card is issued or personalized as shown in Fig 4. The GPB should be stored in a Sector Trailer that is not going to be changed later on. It could be the Sector Trailer of the first sector being used by the application.

Note: This proposal can handle up to 16 cards with the same NUID in the same system.



(1) This flow requires the personalization process to check all the existing System IDs.

Fig 4. How to create a System ID during card personalization

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

© NXP B.V. 2013. All rights reserved.

3.2.2 Double Size UID for card activation, but Single Size NUID for the system

After the card is activated using the Double Size UID, the following proposal can be used to derive a 4 byte NUID out of the 7 byte UID.

Derive NUID out of a Double Size UID

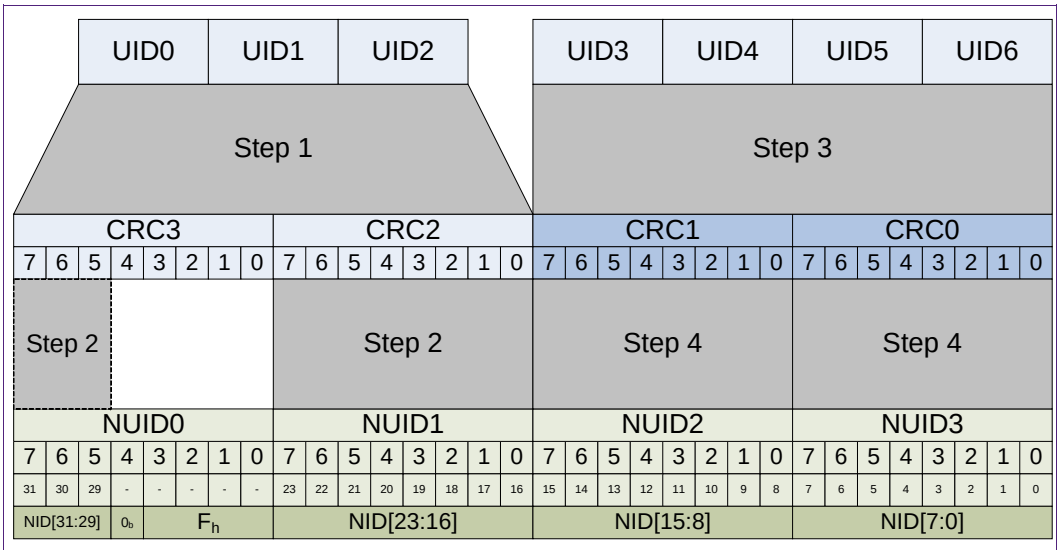
The lower nibble of UID0 must be set to F_h to indicate the non-unique range.

The bit[4] of UID0 shall be set to 0b for compliance reasons.

To generate the 27 bits of the NUID out of the 7 byte UID a CRC calculation shall be done as follows (see Fig 5):

1. Reset the CRC calculator with the standard ISO/IEC 14443 type A preset values: 6363hex.
2. Feed UID0, UID1 and UID2 into the CRC calculator.
3. Result shall be denoted as CRC[3:2]
4. Set NUID[31:29] to CRC[3][7:5] and NUID[23:16] to CRC[2][7:0]
5. Feed UID3, UID4, UID5 and UID6 into the CRC calculator (do not reset the CRC engine before!).
6. Result shall be denoted as CRC[1:0]
7. Set NUID[15:8] to CRC[1][7:0] and NUID[7:0] to CRC[0][7:0]

This mapping ensures that no bit shifting is necessary to build the final NUID from the CRC bytes.



(1) Bit [4..0] of NUID0 must be set to 01111bin.

Fig 5. How to create a single size NUID out of double size UID

This NUID can be treated like the standard NUID.

An example (source code) can be found in Annex C

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

© NXP B.V. 2013. All rights reserved.

3.2.3 RID for card activation, but Single Size NUID for the system

The MIFARE card with RID can be activated like a usual Single Size UID card.

In case RID is enabled, there is a defined and confidential way to retrieve the UID, which then can be used in the (background) system.

If the UID is a Double Size UID, the proposal as shown above (see 3.2.2) can be used to derive a Single Size NUID from the Double Size UID.

3.2.4 RID for card activation, but Double Size UID for the system

The MIFARE card with RID can be activated like a usual Single Size UID card.

In case RID is enabled, there is a defined and confidential way to retrieve the UID, which then can be used in the (background) system.

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

© NXP B.V. 2013. All rights reserved.

3.2.5 MIFARE Classic Authentication

The MIFARE Classic card requires a 4 byte UID input for the authentication command as shown in Table 4.

Table 4. UID bytes as input for the MIFARE Classic Authentication

Table description (optional)

Product	UID	Input for Authentication	Comments
MF1Sxxxx	4 byte UID	4 byte UID (UID0...UID3)	
MF1Sxxxx	4 byte NUID	4 byte NUID (UID0...UID3)	
MF1Sxxxx	7 byte UID	CL2 bytes (UID3...UID6)	
MF1Sxxxx	7 byte UID	CL1 bytes (CT,UID0...UID2)	for shortcut activation
MF1Sxxxx	4 byte RID	4 byte RID (UID0…UID3)	
MF1 PLUS	7 byte UID	CL2 bytes (UID3...UID6)	in SL1 and SL2
MF1 PLUS	4 byte UID	4 byte UID (UID0...UID3)	in SL1 and SL2
MF1 PLUS	4 byte NUID	4 byte NUID (UID0...UID3)	in SL1 and SL2
MF1 PLUS	4 byte RID	-	not available in SL1 or SL2
P5/P6 xxx	4 byte UID	4 byte UID (UID0...UID3)	in B1 / B4 using MIFARE OS
P5/P6 xxx	4 byte NUID	4 byte NUID (UID0...UID3)	in B1 / B4 using MIFARE OS
P5/P6 xxx	7 byte UID	CL2 bytes (UID3...UID6)	in B1 / B4 using MIFARE OS[7]
P5/P6xxx	4 byte RID	4 byte RID (UID0…UID3)	in B1 / B4 using MIFARE OS[7]

3.2.6 Key diversification with MIFARE SAM

The key diversification input must not be the RID. In case of NUID, the 4 bytes NUID can be taken as input.

Refer to [3] for more details.

7. 7 For MIFARE Classic implementation using the MIFARE FleX™ framework

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

© NXP B.V. 2013. All rights reserved.

4. Annex A, Overview over reader UID functionalities

The following tables indicate, how Double Size UID are supported by which reader, reader module or reader IC.

Reader Modules:

Reader	Anti-collision	WRITE	Comment
MF CM200	cascade level 2 possible, but LLL[8] has to be adapted[9]	possible, but LLL has to be adapted	The MF CM200 is not available anymore.
MF CM500	cascade level 2 possible, but LLL has to be adapted[10]	possible, but LLL has to be adapted	The MF CM500 is not available anymore.
PR601	cascade level 2 possible	possible	
PRH601	cascade level 2 possible	possible	

Reader Devices:

Reader	Anti-collision	WRITE	Comment
MF RD260	only cascade level 1, no firmware update or extension possible	only COMPATIBILITY WRITE, no firmware update or extension possible	Does not support 7 Byte UID. The MF RD260 is not available anymore.
MF RD560	only cascade level 1, no firmware update or extension possible	only COMPATIBILITY WRITE, no firmware update or extension possible	Does not support 7 Byte UID. The MF RD560 is not available anymore.

Reader ICs:

Reader	Anti-collision	WRITE	Comment
MF RC171	full cascade level 2 possible, but LLL has to be adapted	possible, but LLL has to be adapted	The MF RC171 is not available anymore.
MFRC500	BFL contains the full cascade level 2 support	BFL contains the full 4 byte WRITE support	
MF RC530	BFL contains the full cascade level 2 support	BFL contains the full 4 byte WRITE support	
MF RC531	BFL contains the full cascade level 2 support	BFL contains the full 4 byte WRITE support	
MF RC630	NXPRdLib contains the full cascade level 2 support	NXPRdLib contains the full 4 byte WRITE support	
MF RC631	NXPRdLib contains the full cascade level 2 support	NXPRdLib contains the full 4 byte WRITE support	
CL RC632	NXPRdLib contains the full cascade level 2	NXPRdLib contains the full 4 byte WRITE	

[8] **L**ow **L**evel **L**ibrary

[9] example see 6.2

[10] example see 6.2

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

© NXP B.V. 2013. All rights reserved.

	support	support	
CL RC663	NXPRdLib contains the full cascade level 2 support	NXPRdLib contains the full 4 byte WRITE support	
MF RC522	BFL/NXPRdLib contains the full cascade level 2 support	BFL/ NXPRdLib contains the full 4 byte WRITE support	
MF RC523	BFL/ NXPRdLib contains the full cascade level 2 support	BFL/ NXPRdLib contains the full 4 byte WRITE support	
PN xxx	BFL/ NXPRdLib contains the full cascade level 2 support	BFL/ NXPRdLib contains the full 4 byte WRITE support	

5. List of References

[1] Doc. No. 0184xx “AN10833 MIFARE Type Identification Procedure”

[2] Doc. No. 1308xx “AN10834 MIFARE ISO/IEC 14443 PICC Selection”

[3] Doc. No. 1653xx “AN10922 Symmetric key diversifications”

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

© NXP B.V. 2013. All rights reserved.

6. Annex B, LLL extension for RC171 and CM220/CM500

6.1 MF RC171 low level library extension: Cascade Anticollision

```
/*****************************************************************************/
int CALL_CONV MfPiccCascAnticoll (unsigned char select_code,
                                  unsigned char bcnt,
                                  unsigned char *snr)
/*****************************************************************************/
{
  int           status;
  unsigned char snr_chk = 0;
  int           i;

  if (MfAssertMode(select_code,0x93|0x95|0x97))
      return (MI_WRONG_PARAMETER_VALUE);

  MfOutp(ENABLE, _PEN | _PRE);         // CRC-disable, Parity enable
  MfOutp(MODE  , __mode);              // __mode preset
  MfOutp(BCNTS ,(unsigned char)(bcnt + 16));           // 16 + number of
bits
  MfOutp(STACON, (unsigned char)(__stacon|_AC));        // anticollision-
mode
  MfDelay50us(4);                                 // BUS-access not allowed
                                                  // for 35us
  MfOutp(DATA, select_code);           // "SELTYPE" of MIFARE1
  MfOutp(DATA, (unsigned char)(((2 + (bcnt >> 3)) << 4) | (bcnt &
0x07)));
                                                   // bytecount higher nibble
                                                   // bitcount  lower  nibble
                                                   // incl. first 2 bytes!!
  for (i = 0; i < (bcnt + 7)/8; i++)
  {
    MfOutp(DATA, snr[i] );
  }
  MfOutp(TOC, TIMEOUT_14443_3); // set timeout
  while (!((status = MfInp(STACON)) & _DV));
  MfOutp(TOC, 0);         // reset timer

  if ((status = MfInp(STACON)) & (_TE | _BE))       // any error
  {
     if (status & _TE)
       return (MI_NOTAGERR);
     if (status & _BE)
     {
       MfDelay50us(10);            // delay 500us
       return (MI_BITCOUNTERR);
     }
  }
  for (i = 0; i < 4; i++)
  {
     snr[i] = MfInp(DATA);
     snr_chk ^= snr[i];
  }
  snr_chk ^= MfInp(DATA);
  // serialnumber check
  if (snr_chk)
    return (MI_SERNRERR);
  return (MI_OK);
}
```


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

© NXP B.V. 2013. All rights reserved.

6.2 MF CM200 / CM500 low level library extension: Cascade Anticollison

```
/*****************************************************************************/
int CALL_CONV MfPiccCascAnticoll (unsigned char select_code,
                                  unsigned char bcnt,
                                  unsigned char *snr)
/*****************************************************************************/
{
  int           status;
  unsigned char snr_chk = 0;
  int           i;

  if (MfAssertMode(select_code,0x93|0x95|0x97))
      return (MI_WRONG_PARAMETER_VALUE);

  MfOutp(ENABLE, _PEN | _PRE);         // CRC-disable, Parity enable
  MfOutp(MODE  , __mode);              // __mode preset
  MfOutp(BCNTS ,(unsigned char)(bcnt + 16));           // 16 + number of
bits
  MfOutp(STACON, (unsigned char)(__stacon|_AC));        // anticollision-
mode
  MfDelay50us(4);                               // BUS-access not allowed
                                                // for 35us
  MfOutp(DATA, select_code);           // "SELTYPE" of MIFARE1
  MfOutp(DATA, (unsigned char)(((2 + (bcnt >> 3)) << 4) | (bcnt &
0x07)));
                                                 // bytecount higher nibble
                                                 // bitcount  lower  nibble
                                                 // incl. first 2 bytes!!
  for (i = 0; i < (bcnt + 7)/8; i++)
  {
    MfOutp(DATA, snr[i] );
  }
  MfOutp(TOC, TIMEOUT_14443_3); // set timeout
  while (!((status = MfInp(STACON)) & _DV));
  MfOutp(TOC, 0);         // reset timer

  if ((status = MfInp(STACON)) & (_TE | _BE))       // any error
  {
     if (status & _TE)
       return (MI_NOTAGERR);
     if (status & _BE)
     {
       MfDelay50us(10);            // delay 500us
       return (MI_BITCOUNTERR);
     }
  }

  for (i = 0; i < 4; i++)
  {
     snr[i] = MfInp(DATA);
     snr_chk ^= snr[i];
  }
  snr_chk ^= MfInp(DATA);
  // serialnumber check
  if (snr_chk)
    return (MI_SERNRERR);
  return (MI_OK);
}
```


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

© NXP B.V. 2013. All rights reserved.

7. Annex C, Source code to derive NUID out of a Double Size UID

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <time.h>
#define BYTE unsigned char

unsigned short UpdateCrc(unsigned char ch, unsigned short *lpwCrc)
{
ch = (ch^(unsigned char)((*lpwCrc) & 0x00FF));
ch = (ch^(ch<<4));
*lpwCrc = (*lpwCrc >> 8)^((unsigned short)ch << 8)^((unsigned
short)ch<<3)^((unsigned short)ch>>4);
return(*lpwCrc);
}

void ComputeCrc(unsigned short wCrcPreset, unsigned char *Data, int
Length, unsigned short &usCRC)
{
unsigned char chBlock;

do {
chBlock = *Data++;
UpdateCrc(chBlock, &wCrcPreset);
} while (--Length);

usCRC = wCrcPreset;

return;
}

void Convert7ByteUIDTo4ByteNUID(unsigned char *uc7ByteUID, unsigned char
*uc4ByteUID)
{
unsigned short CRCPreset = 0x6363;
unsigned short CRCCalculated = 0x0000;
ComputeCrc(CRCPreset, uc7ByteUID, 3,CRCCalculated);
uc4ByteUID[0] = (CRCCalculated >>8)&0xFF;//MSB
uc4ByteUID[1] = CRCCalculated &0xFF; //LSB

CRCPreset = CRCCalculated;
ComputeCrc(CRCPreset, uc7ByteUID+3, 4,CRCCalculated);
uc4ByteUID[2] = (CRCCalculated >>8)&0xFF;//MSB
uc4ByteUID[3] = CRCCalculated &0xFF; //LSB
uc4ByteUID[0] = uc4ByteUID[0]|0x0F;
uc4ByteUID[0] = uc4ByteUID[0]& 0xEF;
}

int main(void)
{
       int i;
             unsigned char uc7ByteUID[7] =
{0x04,0x18,0x3F,0x09,0x32,0x1B,0x85};//4F505D7D

       unsigned char uc4ByteUID[4] = {0x00};

       Convert7ByteUIDTo4ByteNUID(uc7ByteUID,uc4ByteUID);
       printf("7-byte UID = ");
       for(i = 0;i<7;i++)
             printf("%02X",uc7ByteUID[i]);
```


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

© NXP B.V. 2013. All rights reserved.

```
        printf("\t4-byte FNUID = ");
        for(i = 0;i<4;i++)
              printf("%02X",uc4ByteUID[i]);
              getch();
return(0);
}
```


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

© NXP B.V. 2013. All rights reserved.

8. Legal information

8.1 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.

8.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the *Terms and conditions of commercial sale* of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

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.

8.3 Licenses

ICs with DPA Countermeasures functionality

NXP ICs containing functionality implementing countermeasures to Differential Power Analysis and Simple Power Analysis are produced and sold under applicable license from Cryptography Research, Inc.

8.4 Trademarks

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

MIFARE, MIFARE Ultralight, DESFire, MIFARE Plus, MIFARE FleX, SmartMX — are trademarks of NXP B.V.

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

© NXP B.V. 2013. All rights reserved.

9. Contents

1. Introduction 3
2. MIFARE and ISO/IEC 14443 UIDs 3
2.1 Single Size UID 5
2.1.1 Random ID (RID) 5
2.1.2 Fixed but non-unique ID (FNUID) 6
2.1.3 Re-used UID (ONUID) 6
2.2 Double Size UID 6
2.2.1 Manufacturer Code 6
2.2.2 Unique ID ranges for Double Size UIDs 6
2.3 Triple Size UID 6
2.4 UID used in MIFARE products 7
3. UID and MIFARE Classic / MIFARE Plus 8
3.1 Card activation 8
3.1.1 Single Size NUID 8
3.1.2 Double Size UID with “shortcut” 8
3.1.3 RID 8
3.2 UID in the contactless system 9
3.2.1 Single Size NUID for card activation and for the system 9
3.2.2 Double Size UID for card activation, but Single Size NUID for the system 11
3.2.3 RID for card activation, but Single Size NUID for the system 12
3.2.4 RID for card activation, but Double Size UID for the system 12
3.2.5 MIFARE Classic Authentication 13
3.2.6 Key diversification with MIFARE SAM 13
4. Annex A, Overview over reader UID functionalities 14
5. List of References 15
6. Annex B, LLL extension for RC171 and CM220/CM500 16
6.1 MF RC171 low level library extension: Cascade Anticollision 16
6.2 MF CM200 / CM500 low level library extension: Cascade Anticollison 17
7. Annex C, Source code to derive NUID out of a Double Size UID 18
8. Legal information 20
8.1 Definitions 20
8.2 Disclaimers 20
8.3 Licenses 20
8.4 Trademarks 20
9. Contents 21

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

© NXP B.V. 2013. **All rights reserved.**

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

Date of release: 02 October 2013
190731
Document identifier: AN10927