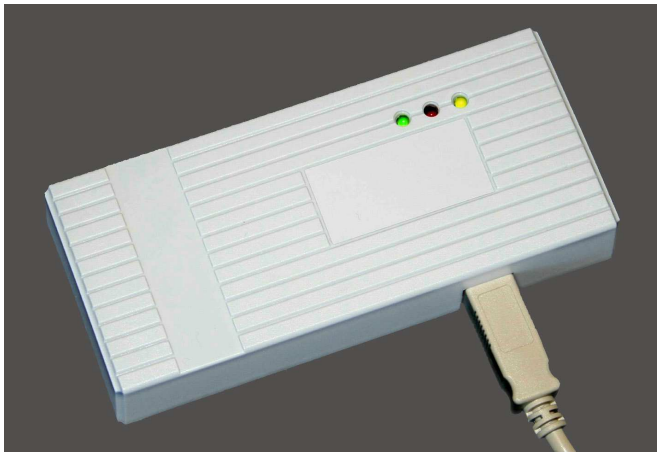
727-52 REV4 USB MIFARE® Reader Writer Module

Data sheet

Overview



The 727-52 multi sector MIFARE® Reader/Writer is an intelligent 13.56MHz contactless reader writer designed to operate with Mifare® Std 1k/4k, Mifare PlusX/PlusS (in security level 1), Mifare Ultralight cards or NTAG NFC tags.

The reader/writer uses a PHILIPS MIFARE® reader IC and complies to ISO 14443A.

The multi sector MIFARE® Reader/Writer can read and write to any sector and block on the card and is MAD (Mifare application directory) compliant. All the encryption needed to provide a secure and reliable transaction is handled automatically with a few simple commands.

Read/write of MIFARE® Plus cards, both the X and S types, is supported in compatibility mode (SL1) which offers a further level of security with the option of 128 bit AES authentication. In addition MIFARE® Plus cards may be personalised from SL0 to any of SL1, SL2 or SL3 states, this includes programming any or all the AES keys that may be required.

A USB interface is used to communicate between the unit and a PC. When plugged into the PC the device enumerates as a virtual serial port.

The reader has three independent LEDs and a beeper which are controlled via commands.

Features and specifications

- Reads MIFARE® cards unique ID number, UID
- Reads and writes to MIFARE® Ultralight pages[1] and NTAG NFC tags
- Reads data from any sector block on a MIFARE® Std 1k or 4k card or MIFARE PLUS card
- Writes[2] data to any sector block on a MIFARE® Std 1k or 4k card or MIFARE PLUS card
- Reader can store 32 keys for use as A or B keys
- Reader can store 16 128 bit AES keys for Mifare Plus
- Supports optional 128 bit AES authentication of Mifare Plus cards to prevent card copying and tampering.
- Supports MAD1 and MAD2
- Commands to support transactions using the MIFARE® VALUE[3] format
- Commands to support the personalisation and storing of AES keys on MIFARE® Plus SL0 cards to any of the security levels, SL1, SL2 or SL3
- Power requirements (provided by the USB host): 5V dc. 100 mA.
- RF Frequency: 13.56 MHz.
- Piezzo sounder emits a 4kHz tone.
- Typical reading range - 20mm to 50mm, dependant on card type.
- Communication interface: USB 2.0 - when connected to the PC the device enumerates as a standard COM port by using the standard Windows driver “usbser.sys” for CDC class USB devices.
- USB connector: type B
- Operating temperature range: 0ºC - +50ºC

- Weight: 185 grams
- Dimensions: 118 x 54 x 21mm

Note 1. Not all pages are available to the user eg. Mifare Ultralight pages 0 and 1 are read only, page 3 is one-time Programmable.
Note 2. The manufacturer's data, sector 0 block 0, is read only.
Note 3. The MIFARE® VALUE format is a fixed data format for a sector block that permits error detection, correction and backup management. The size of the data is a signed 4 byte value, for reasons of data integrity and security a value is stored 3 times. A 1 byte address, stored 4 times is also stored in the sector block and is used when implementing backup management.

Command format

Commands and replies are sent in ASCII and are case sensitive, only keyboard characters are used. Data is sent as ascii chars. A terminal program (HyperTerminal) can be used to send commands and receive replies.

Two COMMAND FORMATS are provided for, one with an appended checksum and one without, either may be used

With checksum the format is:

<$><A>,<C>,<d1>,<d2>,....,<CHK><CR><LF>

Without checksum the format is:

<!><A>,<C>,<d1>,<d2>,....><CR><LF>

where,

<$> command header that indicates start of the command with checksum
<!> command header that indicates start of the command without checksum
<A> is a 1 digit target address code in decimal, this is always 1 for the Reader Writer
<C> is one or more command identification characters
<d1> <d2> ... are optional command parameters in decimal (or hex if preceded by 0x)
<CHK> is the checksum, where sent,
checksum formatted as ‘c-style’ hex number e.g. 0xA3
<CR> is a carriage return \r <enter>
<LF> is a line feed \n - optional and is ignored

the ',' is used as a delimiter between characters

The checksum is calculated by performing an 8 bit addition of the ascii value of each char making up the command and includes the start char and comma delimiters but not the checksum itself.

REPLY FORMAT

The reply always includes a checksum, the format is:

<$><A>,<REPLY>,<y..y>....,<CHK><CR><LF>

where,

<$> reply header that indicates start of the reply
<A> is a 1 digit target address code in decimal, this is always 0 for the application hardware
<REPLY> is the reply, may be, OK for a successfully implemented command, ERROR xx for any failure in deciphering or executing the command (where xx is the error code) or a command identification character
<y...y> any parameters used by the reply
<CHK> is the checksum,
checksum formatted as ‘c-style’ hex number e.g. 0x0B
<CR> is a carriage return \r <enter>
<LF> is a line feed \n - optional and is ignored

the ',' is used as a delimiter between characters

The checksum is calculated by performing an 8 bit addition of the ascii value of each char making up the command and includes the start char and comma delimiters but not the checksum itself.

Note. Where a hex value is given or requested, two chars always follow the 0x
Example - for hex value 1, 0x1 is illegal, it is required to be 0x01

The reader/writer has a buffer which is cleared and starts filling after the header has been received, the command is processed after a <CR> is received. All commands are replied to.

Command Set

Note. For ease of reading the commands and replies are shown here without the start character, address code, checksum and <CR><LF>

An example of the use of each command is given together with the reply for both the with and without checksum format.

Note. Whilst these examples are real commands that can be issued the replies are dependant upon the information on the card which may differ from the users test card.

Query Commands

Return Version Information
I

Function. Outputs both hardware and firmware version information
Reply list: 1. Up to 21 ascii chars of version information, example 727-52 v2.01 build ab
2. ERROR xx

Example, without checksum

```
!1,I
$0,727-52 v2.01 build ab,0x4A
```

Example, with checksum

```
$1,I,0xF6
$0,727-52 v2.01 build ab,0x4A
```

Control Commands

Bootloader Command
L
Reply List: 1. OK
2. ERROR xx

Example, with checksum

```
$1,L,0xF9
$0,OK,0x46
```

To prevent unintentionally entering the bootloader mode and subsequently erasing the current firmware in flash this command will only be accepted by the reader/writer if sent with the checksum.

Note. Once this command has been accepted the current firmware in flash has been erased and

no communication is now possible. The reader/writer will remain in this mode until new firmware has been downloaded by using proprietary bootloader software.

Control Beeper
B,tttt

Function: Turns on the beeper for tttt ms
where: tttt is the beeper on time in milliseconds entered as decimal ascii chars, will accept any time from 0ms to 9999ms

example B,1 gives 1ms beep time
B,0 turns beeper off
Reply list: 1. OK
2. ERROR xx

Example, beep for 100ms, without checksum

```
!1,B,100
$0,OK,0x46
```

Example, beep for 100ms, with checksum

```
$1,B,100,0xAC
$0,OK,0x46
```

Control RF Field
F,x

Function. Turns RF field either on or off
where: x is RF field state
0=OFF
1=ON

Reply list: 1. OK
2. ERROR xx

Example, turn off RF field, without checksum

```
!1,F,0
$0,OK,0x46
```

Example, turn on RF field with checksum

```
$1,F,1,0x50
$0,OK,0x46
```

Control Green LED
G,x

Function. Turns Green LED either on or off
where: x is Green LED state
0=OFF
1=ON

Reply list: 1. OK
2. ERROR xx

Example, turn on GREEN LED, without checksum

```
!1,G,1
$0,OK,0x46
```

Example, turn off GREEN LED, with checksum

```
$1,G,0,0x50
$0,OK,0x46
```

Control Red(Scarlet) LED
S,x

Function. Turns Red LED either on or off
where: x is Red LED state
0=OFF
1=ON

Reply list: 1. OK
2. ERROR xx

Example, turn on RED LED, without checksum

```
!1,S,1
$0,OK,0x46
```

Example, turn off RED LED, with checksum

```
$1,S,0,0x5C
$0,OK,0x46
```

Control Yellow LED
Y,x

Function. Turns Yellow LED either on or off
where: x is Yellow LED state
0=OFF
1=ON

Reply list: 1. OK
2. ERROR xx

Example, turn off YELLOW LED, without checksum

```
!1,Y,0
$0,OK,0x46
```

Example, turn on YELLOW LED, with checksum

```
$1,Y,1,0x63
$0,OK,0x46
```

Transaction Commands

Read and Return UID
U

Function. For any MIFARE® card in the field this command will return the card's UID. For cards with a 4 byte UID (MIFARE® STD) 4 bytes are returned, for cards with a 7 byte UID 7 bytes are returned.
Reply List: 1. 4 or 7 byte UID in hex, in 'reverse' order. Note. Reverse order is opposite to that given by the Philips PEGODA reader, but is logically the correct order.
2. ERROR xx

Example, without checksum (Mifare Std 4 byte UID)

```
!1,U
$0,436E37F2,0x70
```

Example, with checksum (Mifare Plus 7 byte UID)

```
$1,U,0x02
$0,802861A91F6004,0xA0
```

Check Card Type
PT

Function. Checks card type by activating the card which then replies with a 2 byte code

Reply List:

1. 2 byte code in hex (see table 1)
2. ERROR xx

Code	Card
0x00	Mifare Ultralight NTAG NFC tag
0x08	Mifare Std 1k Mifare Plus S (SL1)
0x18	Mifare Std 4k Mifare PlusX (SL1)
0x20	Desfire Mifare PlusS (SL0) Mifare PlusX (SL0)
0x11	Mifare PlusX (SL2)

Table 1. Mifare card type reply codes

Note. The code is not unique for each type of card and therefore not all cards can be unambiguously identified.

Example, check a Mifare Std 1k card, without checksum

```
!1,PT
$0,0x08,0xBC
```

Example, check a Mifare PlusX SL1 card, with checksum

```
$1,PT,0x51
$0,0x18,0xBD
```

Write Key
K,ii,0xhhhhhhhhhhhh

Function: ii is an index to where the key is stored, 0xhhhhhhhhhhhh is the 6 byte key in hex. Up to 32 keys may be stored, these are the A and B key pairs for each sector. This key is written to a secure read only area in memory. This key cannot be read back after being loaded but new keys may be loaded and will overwrite the existing keys.

How the keys are indexed is up to the user but for a read or write transaction to be successful the correct key type (A or B) and index needs to be used with the relevant command.

where ii is the key storage index

range 00 < ii < 31

Reply list: 1. OK
2. EEROR xx

Example, store the transport key 0xFFFFFFFFFFFF in index position 00, without checksum

```
!1,K,00,0xFFFFFFFFFFFF
$0,OK,0x46
```

Example, store a proprietary key 0x123456789012 in index position 01 with checksum

```
$1,K,01,0x123456789012,0xC9
$0,OK,0x46
```

Note. Keys provide the security for the MIFARE system and the user must protect their keys, therefore secure keys should be written only once, preferably at the factory.

Write AES Authentication Key
PK,ii,0xhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhh

Function: ii is an index to where the AES key is stored, 0xhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhh is the 16 byte (128 bits) key in hex. Up to 16 keys may be stored. This key cannot be read back after being loaded but new keys may be loaded and will overwrite the existing keys.

How the keys are indexed is up to the user but for an optional AES authentication to be successful the correct index needs to be used when using the AES Authentication command.

where ii is the key storage index

range 00 < ii < 16
Reply list: 1. OK
2. EEROR xx

Example, store the AES key 0xCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC in index position 01, without checksum

```
!1,PK,01,0xCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
$0,OK,0x46
```

Example, store the AES key 0x12345678901234567890123456789012 in index position 02 with checksum

```
$1,PK,02,0x12345678901234567890123456789012,0x34
$0,OK,0x46
```

Note. Keys provide the security for the MIFARE system and the user must protect their keys, therefore keys should be written only once, preferably at the factory.

AES Authentication

PA,ii

Function. Does an AES authentication on the card in th field using the key found at the AES key index ii

where ii is the key index in decimal
range 00 ≤ ii ≤ 16

Reply List:
1. OK
2. ERROR xx

Example, authenticate a Mifare Plus card using key found at index position 01, without checksum

```
!1,PA,01
$0,OK,0x46 (successful authetication)
or
$0,ERROR 3,0xB9  (authentication failed)
```

Example, authenticate a Mifare Plus card using key found at index position 02, with checksum

```
$1,PA,02,0xCC
$0,OK,0x46 (successful authetication)
or
$0,ERROR 3,0xB9  (authentication failed)
```

Read Data Sector Block
R,ss,bb,k,ii

Function. Reads 16 bytes of data from the addressed sector block ss bb of the MIFARE® card, uses the given key type k found at the key index ii

where ss is the sector number in decimal
range 00 ≤ ss ≤ 15 for MIFARE® 1k, 2k and 4k cards
range 16 ≤ ss ≤ 31 for MIFARE® 2k and 4k cards
range 32 ≤ ss ≤ 39 for MIFARE® 4k cards

where bb is the block number in decimal
range 00 ≤ bb ≤ 03 for MIFARE® 1k, 2k and 4k cards
range 04 ≤ bb ≤ 15 for MIFARE® 4k cards

where k is the key type
range k=A or k=B

where ii is the key index in decimal
range 00 ≤ ii ≤ 31

Reply List:
1. R,ss,bb,0xhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhh (a 16 byte hex number)
2. ERROR xx

Example, read the data from sector 1 block 0 using key A found at index position 01, without checksum

```
!1,R,01,00,A,01
$0,R,01,00,0x01000000000000000000000000000000,0xEC
```

Example, read the data from sector 1 block 1 using key A found at index position 01, with checksum

```
$1,R,01,01,A,01,0x13
```

```
$0,R,01,01,0x0101000000000000000000000000000,0xEE
```

Return AID Sector number
MS,0xaaaa

Function. Returns the sector number addressed by the AID 0xaaaa, follows MAD rules (MAD1 and MAD2 are supported)

where 0xaaaa is the 2 byte AID in hexadecimal

Reply List:
1. MS,ss
2. ERROR xx

where ss is the sector number in decimal, pointed to by the AID
range 00 ≤ ss ≤ 15 for MIFARE® 1k, 2k and 4k cards
range 16 ≤ ss ≤ 31 for MIFARE® 2k and 4k cards
range 32 ≤ ss ≤ 39 for MIFARE® 4k cards

Example, return the sector number from the given AID, without checksum

```
!1,MS,0x4702
$0,MS,07,0xDF
```

Example, return the sector number from the given AID, without checksum

```
$1,MS,0x4702,
$0,MS,07,0xEE
```

Read Ultralight/NTAG Page Data
TR,ppp

Function. Reads 16 bytes of data starting from the addressed page ppp on a MIFARE® Ultralight card or NTAG NFC tag,

where ppp is the page number in decimal
range 00 ≤ ppp ≤ 230 for MIFARE® Ultralight cards and NTAG NFC tags *(for backward compatibility to versions before 3.1 the parameter ppp only requires 2 digits for pages up to and including 99)*

Reply List:
1. R,ppp,00,0xhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhh (a 16 byte hex number)
2. ERROR xx

Example, read the data from page 4, without checksum

```
!1,TR,04
$0,R,04,00,0x44444444555555556666666677777777,0x9E
```

Example, read the data from page 5, with checksum

```
$1,TR,05,0xE4
$0,R,05,00,0x55555555666666667777777788888888,0xBF
```

Write Data Sector Block

W,ss,bb,k,ii,0xhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhh

Function. Writes up to 16 bytes of data to the addressed sector block ss, bb of the MIFARE® card using the given key type k found at the key index ii. The data must be at least 1 byte and no more than 16 bytes long. Unused bytes in the block will be filled with NUL's (0x00).

where ss is the sector number in decimal
range 00 ≤ ss ≤ 15 for MIFARE® 1k, 2k and 4k cards
range 16 ≤ ss ≤ 31 for MIFARE® 2k and 4k cards
range 32 ≤ ss ≤ 39 for MIFARE® 4k cards

where bb is the block number in decimal
range 00 ≤ bb ≤ 03 for MIFARE® 1k, 2k and 4k cards
range 04 ≤ bb ≤ 15 for MIFARE® 4k cards

where k is the key type
range k=A or k=B

where ii is the key index in decimal
range 00 ≤ ii ≤ 31

Data is in hexadecimal

Reply List: 1.OK
2. ERROR xx

Example, write the data to sector 4 block 1 using key A found at index position 01, without checksum

```
!1,W,04,01,A,01,0x0123456789ABCDEFFEDCBA9876543210
$0,OK,0x46
```

Example, write the data to sector 4 block 2 using key A found at index position 01, with checksum

```
$1,W,04,02,A,01,0x04020000000000000000000000000000,0xF6
$0,OK,0x46
```

Warning. All blocks including the sector trailer can be written to. Writing data with the incorrect format to the sector trailer will permanently destroy read or write access to the whole sector.

Write Ultralight/NTAG Page Data
TW,ppp,0xhhhhhhhh

Function. Writes up to 4 bytes of data to the addressed page ppp of the MIFARE® Ultralight card or NTAG NFC tag. The data must be at least 1 byte and no more than 4 bytes long. Unused bytes in the block will be filled with NUL's (0x00).

where ppp is the page number in decimal
range 00 ≤ ppp ≤ 230 for MIFARE® Ultralight cards and NTAG NFC tags *(for backward compatibility to versions before 3.1 the parameter ppp only requires 2 digits for pages up to and including 99)*

Data is in hexadecimal

Reply List: 1.OK
2. ERROR xx

Example, write the data to page 4, without checksum

```
!1,TW,04,0x44444444
$0,OK,0x46
```

Example, write the data to page 5, with checksum

```
$1,TW,05,0x55555555,0x65
$0,OK,0x46
```

Note. Page 0, page 1 and the first byte of page 2 are manufacturer's data and are write protected.

Caution. Page 2 contains lock bits which can be used to lock program memory to read only. Page 3 is the OTP page where the bits may only be 'set' once.

Read VALUE² Sector Block
V,ss,bb,k,ii

Function. Reads 4 bytes of data in the MIFARE® VALUE² format from the addressed sector block ss, bb of the MIFARE® card, uses the given key type k found at the key index ii

where ss is the sector number in decimal
range $00 \leq ss \leq 15$ for MIFARE® 1k, 2k and 4k cards
range $16 \leq ss \leq 31$ for MIFARE® 2k and 4k cards
range $32 \leq ss \leq 39$ for MIFARE® 4k cards

where bb is the block number in decimal
range $00 \leq bb \leq 02$ for MIFARE® 1k and 4k cards
range $04 \leq bb \leq 14$ for MIFARE® 4k cards

where k is the key type
range k=A or k=B

where ii is the key index in decimal
range $00 \leq ii \leq 31$

Reply List:
1. V,ss,bb,0xhhhhhhhh(a 4 byte signed hex number)
2. ERROR xx

Example, read the VALUE from sector 5 block 0 using key A found at index position 01, without checksum

```
!1,V,05,00,A,01
$0,V,05,00,0x00100000,0x74
```

Example, read the VALUE from sector 5 block 1 using key A found at index position 01, with checksum

```
$1,V,05,01,A,01,0x1B
$0,V,05,01,0x00100000,0x75
```

Decrement VALUE² Sector Block
D,ss,bb,k,ii,0xhhhhhhhh

Function. The 4 bytes of hex data will be subtracted from the VALUE² found at the addressed sector block, ss,bb, of the MIFARE® card using the given key type k found at the key index ii.

where ss is the sector number in decimal
range 00 ≤ ss ≤ 15 for MIFARE® 1k, 2k and 4k cards
range 16 ≤ ss ≤ 31 for MIFARE® 2k and 4k cards
range 32 ≤ ss ≤ 39 for MIFARE® 4k cards

where bb is the block number in decimal
range 00 ≤ bb ≤ 02 for MIFARE® 1k, 2k and 4k cards
range 04 ≤ bb ≤ 14 for MIFARE® 4k cards

where k is the key type
range k=A or k=B

where ii is the key index in decimal
range 00 ≤ ii ≤ 31

where 0xhhhhhhhh is a 4 byte signed hex number, negative numbers are not permitted (this effectively limits numbers to 31 bits).

Reply List: 1.OK
2. ERROR xx

Example, decrement by the amount 0x00000001 the VALUE at sector 5 block 0 using key A found at index position 01, without checksum

```
!1,D,05,00,A,01,0x00000001
$0,OK,0x46
```

Example, decrement by the amount 0x00000001 the VALUE at sector 5 block 1 using key A found at index position 01, with checksum

```
$1,D,05,01,A,01,0x00000001,0x5E
$0,OK,0x46
```

Increment VALUE[2] Sector Block
A,ss,bb,k,ii,0xhhhhhhhh

Function. The 4 bytes of hex data will be added to the VALUE[2] found at the addressed sector block, ss,bb, of the MIFARE® card using the given key type k found at the key index ii.

where ss is the sector number in decimal
range 00 ≤ ss ≤ 15 for MIFARE® 1k, 2k and 4k cards
range 15 ≤ ss ≤ 31 for MIFARE® 2k and 4k cards
range 32 ≤ ss ≤ 39 for MIFARE® 4k cards

where bb is the block number in decimal
range 00 ≤ bb ≤ 03 for MIFARE® 1k, 2k and 4k cards
range 04 ≤ bb ≤ 15 for MIFARE® 4k cards

where k is the key type
range k=A or k=B

where ii is the key index in decimal
range 00 ≤ ii ≤ 31

where 0xhhhhhhhh is a 4 byte signed hex number, negative numbers are not permitted (this effectively limits numbers to 31 bits).

Reply List: 1.OK

2. ERROR xx

Example, increment by the amount 0x00000001 the VALUE at sector 5 block 0 using key A found at index position 01, without checksum

```
!1,A,05,00,A,01,0x00000001
$0,OK,0x46
```

Example, increment by the amount 0x00000001 the VALUE at sector 5 block 1 using key A found at index position 01, with checksum

```
$1,A,05,01,A,01,0x00000001,0x5B
$0,OK,0x46
```

Write VALUE² Sector Block
X,ss,bb,k,ii,0xhhhhhhhh

Function. Writes 4 bytes of hex data in the MIFARE® VALUE² format to the addressed sector block, ss,bb, of the MIFARE® card using the given key type k found at the key index ii.

where ss is the sector number in decimal
range $00 \leq ss \leq 15$ for MIFARE® 1k, 2k and 4k cards
range $15 \leq ss \leq 31$ for MIFARE® 2k and 4k cards
range $32 \leq ss \leq 39$ for MIFARE® 4k cards

where bb is the block number in decimal
range $00 \leq bb \leq 03$ for MIFARE® 1k, 2k and 4k cards
range $04 \leq bb \leq 15$ for MIFARE® 4k cards

where k is the key type
range k=A or k=B

where ii is the key index in decimal
range $00 \leq ii \leq 31$

where 0xhhhhhhhh is a 4 byte signed hex number, negative numbers are not permitted (this effectively limits numbers to 31 bits).

Reply List: 1.OK
2. ERROR xx

Example, write the amount 0x00100000 as the VALUE at sector 5 block 0 using key A found at index position 01, without checksum

```
!1,X,05,00,A,01,0x00100000
$0,OK,0x46
```

Example, write the amount 0x00100000 as the VALUE at sector 5 block 1 using key A found at index position 01, with checksum

```
$1,X,05,01,A,01,0x00100000,0x72
$0,OK,0x46
```

Note: The manufacturer's block, sector 0 block 0 and all the sector trailer's, block 3, cannot be used in the MIFARE® VALUE² format.

Note. An OK reply does not necessarily guarantee that the transaction took place successfully, it is up to the user application to validate this.

Personalisation Commands

Write SLO Block

PW,0xaaaa,0xhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhh

Function. For only MIFARE® Plus cards in the SL0 state writes 16 bytes of data, 0xhh..hh to the SL0 block addressed by 0xaaaa. Must be strictly 16 bytes of data.
Reply List: 1. OK.
2. ERROR xx

Example, without checksum

```
!1,PW,0x9000,0xAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
$0,OK,0x46
```

Example, with checksum

```
$1,PW,0x9001,0xBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB,0x06
$0,OK,0x46
```

Write Commit Perso

PC

Function. For only MIFARE® Plus cards in the SL0 state (with the minimum required AES keys programmed) writes a Commit Perso command so that the card enters the SL1 state.
Reply List: 1. OK.
2. ERROR xx

Example, without checksum

```
!1,PC
$0,OK,0x46
```

Example, with checksum

```
!$1,P,0x40
$0,OK,0x46
```

Error Codes

The table shows the error codes returned when a transaction with a MIFARE® card fails

Error Code	Error Message
1	No card in the field
2	Communication error
3	Authentication error
4	Corrupt Value
5	Transaction value negative
6	Transaction failed
7	Command format error
8	MAD transaction error