723-52 REV5

Multi Sector MIFARE® Reader/Writer Module

Data Sheet

General Description

The 723-52 multi sector MIFARE® Reader/Writer Module is a fully encapsulated device containing all the electronics required to perform a read or write on a Mifare® Std 1k, 4k, Mifare PlusX/PlusS (in security level 1 mode), Ultralight card or NTAG NFC tag with just the addition of an external antenna.

The module can read and write to any sector and block on the card and is MAD compliant. All the encryption needed to provide a secure and reliable transaction is handled automatically with a few simple commands.

Mifare Plus cards, both the X and S types can be personalised and operated in compatibility mode (SL1) which offers a further level of security with the option of 128 bit AES authentication.

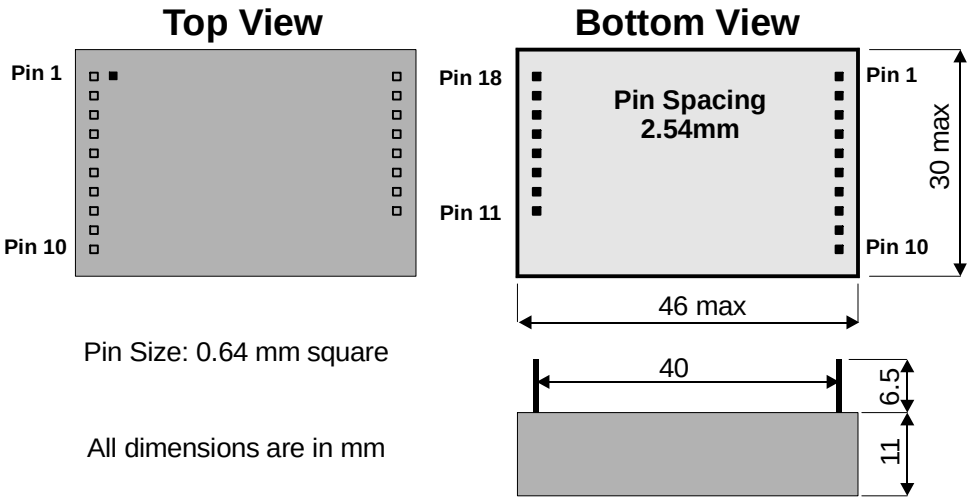
An RS232 TTL level interface is used to communicate between the module and application hardware.

Three external LEDs and a beeper can be connected to form a complete interactive reader.

Features

- 5V to 12V operating voltage, current consumption typically 100mA
- Matched for 50 ohm 13.56MHz external antenna
- Contactless interface to specification ISO/IEC 14443A
- Reads data from any sector block on a MIFARE® Std 1k, 4k card or MIFARE PLUS card or page on a MIFARE® Ultralight card or NTAG NFC tag
- Writes data to any sector block on a MIFARE® Std 1k, 4k card or MIFARE PLUS card or page on a MIFARE® Ultralight card or NTAG NFC tag
- 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
- Supports MAD1 and MAD2
- Commands to support transactions using the MIFARE® VALUE format
- Commands for full personalisation of Mifare Plus cards
- RS232 (TTL levels) serial communication port at 19200 baud
- Red, Yellow and Green LED drivers and control
- 4kHz beeper driver and control
- Read range from 20 to 100 mm dependant on external antenna and card type
- Compact size 30 x 46 x 11 mm, weight 30g
- Operating temperature range -20^{O}C to +60^{O}C
- Uses a Philips MIFARE® reader IC

Pinout Description & Dimensions



Pin #	Pin name	Function
1	RESERVED	Reserved - do not connect.
2	RESERVED	Reserved - do not connect.
3	RESERVED	Reserved - do not connect.
4	TTL RX[1]	RS232 TTL level Rx.
5	RESERVED	Reserved - do not connect.
6	TTL TX[1]	RS232 TTL level Tx.
7	RED LED[3]	RED LED drive, connect to anode and ground cathode.
8	YELLOW LED[3]	YELLOW LED drive, connect to anode and ground cathode.
9	GREEN LED[3]	GREEN LED drive, connect to anode and ground cathode.
10	RESERVED	Reserved - do not connect.
11	ANT GND[2]	External antenna GND connection.
12	ANT TX	External antenna connection.
13	ANT GND[2]	External antenna GND connection.
14	RESERVED	Reserved - do not connect.
15	BEEPER	Open collector output for connection to an external piezo beeper.
16	NC	No connect
17	VCC[4]	Connect +5V to +12V from power supply.
18	0V	Connect 0V from power supply.

Note 1: TTL TX connects to the internal microprocessor UART and the output is 5V TTL levels suitable for connecting to the application microprocessor UART, if EIA compliant RS232 outputs are required then connect this to a 232 driver IC.

Note 2: These antenna GND's are internally connected, one or both may be connected to the external antenna GND.

Note 3: Includes internal current limiting resistor so no external resistor is required.

Note 4: For voltages less than 5.5V the internal regulator drops out of regulation and any ripple on the external supply is passed through to the module, in these cases it is recommended that the supply ripple be less than 20mVpp.

Antenna

The module requires an external H field antenna resonant at 13.56MHz and matched to 50 ohms. This usually takes the form of a multiturn (typically 3 turns) inductive loop with parallel resonant capacitor and an 'L' match to 50 ohms. See the application note 'Designing an antenna for the Mifare module' for more information or contact the supplier to supply an antenna to your specification.

RS232 Communication

There is a full duplex serial communication (TTL levels) connection to the Reader Writer module, which connects directly to the application microprocessor UART or if EIA compliant RS232 is required may be connected to a 232 driver IC.

The baud rate is 19200. Data format is 8 bits, no parity, and 1 stop bit.

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

Note. The delimeters < > are not sent with the command, they indicate that some other 'char' is substituted here.

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

Note. The delimeters < > are not sent with the command, they indicate that some other 'char' is substituted here.

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 20 ascii chars of version information, example 723-52 v2.00
2. ERROR xx

Example, without checksum

```
!1,I
$0,723-52 v2.00dk,0x01
```

Example, with checksum

```
$1,I,0xF6
$0,723-52 v2.00dk,0x01
```

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.

Software Reset (Clear)
C

Function. Initiates a microprocessor software reset
Reply list: 1. OK (before resetting)
2. ERROR xx

Example, without checksum

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

Example, with checksum

```
$1,C,0xF0
$0,OK,0x46
```

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 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

```
!1,U
$0,11EA7C52,0x75
```

Example, with checksum

```
$1,U,0x02
$0,11EA7C52,0x75
```

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
```

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. This is a Mifare Plus function and is used to store up to 16 128 bit AES keys in the reader, which can then be referenced by index for Mifare Plus authentication

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 the field using the key found at the AES key index ii

where ii is the key index in decimal
range $00 \leq ii \leq 16$

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 authentication)
or
$0,ERROR 3,0xB9  (authentication failed)
```

Note. This is a Mifare Plus function and for cards that have been personalised with an SL1 card authenthentication key this command can be used prior to reading or writing sector data to verify that the card is not a clone.

Write PERSO Key
PW,0xaaaa,0xhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhh

Function: writes the 16 byte (128 bit) AES key 0xhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhh to the Mifare Plus key address 0xaaaa.

where 0xaaaa is the 2 byte Mifare Plus key address
range - only certain values are valid - see Appendix A

Reply list: 1. OK
2. EEROR xx

Example, write the AES key 0x12345678901234567890123456789012 as the Mifare Plus Master Key (address 0x9000), without checksum

```
!1,PW,0x9000,0xF1F2F3F4F5F61A2A3A4A5A6AE0E9E8E7
$0,OK,0x46
```

Example, write the AES key 0x1A1B1C1D1E1F2A2B2C2D2E2F3A3B3C3D as the Mifare Plus Card Configuration Key (address 0x9001), with checksum

```
$1,PW,0x9001,0x1A1B1C1D1E1F2A2B2C2D2E2F3A3B3C3D,0x18
$0,OK,0x46
```

Note. This is a Mifare Plus function and is used to personalise a Mifare Plus card

---

Write Commit PERSO Key
PC

Function: writes the Commit PERSO command

Reply list: 1. OK
2. EEROR xx

Example, write the Commit PERSO Key, without checksum

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

Example, write the Commit PERSO Key, with checksum

```
$1,PC,0x40
$0,OK,0x46
```

Note. This is a Mifare Plus function and is used to Commit the personalisation data to a Mifare Plus card. Can only follow a Write PERSO Key command (at least 4 keys must have been written). If successful then a Mifare Plus card in SL0 state will move to SL1, this operation is irreversible!

Note. To move an SL1 card to SL2 (or SL2 to SL3) then use the Write PERSO Key command with the Level 2 (or 3) Switch Key at the relevant address

---

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[â] 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 16 ≤ 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,01,01,0x01010000000000000000000000000000,0xEE
```

Read Data AID Sector Block
MR,0xaaaa,bb,k,ii

Function. Reads 16 bytes of data from the sector block addressed by the AID 0xaaaa and the block bb, follows MAD rules (MAD1 and MAD2 are supported), uses the given key type k found at the key index ii

where 0xaaaa is the 2 byte AID in hexadecimal

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

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, read the data from sector addressed by the AID 0x0801 at block 0 using key A found at index position 01, without checksum

```
!1,MR,0x0801,00,A,01
$0,R,03,00,0x,0x08010000000000000000000000000000,0xF6
```

Example, read the data from sector addressed by the AID 0x1003 at block 0 using key A found at index position 01, with checksum

```
$1,MR,0x1003,00,A,01,0x6A
$0,R,09,00,0x09000000000000000000000000000000,0xFC
```

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 4.1 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 31 ≤ ss ≤ 39 for MIFARE® 4k cards

where bb is the block number in decimal
range 00 ≤ bb ≤ 03 for MIFARE® 1k 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,01,0x040200000000000000000000000000,0xF6
$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 Data AID Sector Block
MW,0xaaaa,bb,k,ii,0xhhhhhhhhhhhhhhhhhhhhhhhhhhhhhh

Function. Writes up to 16 bytes of data to the sector block addressed by the AID 0xaaaa and block bb of the MIFARE® card, uses 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 0xaaaa is the 2 byte AID in hexadecimal

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

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

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

Data is in hexadecimal

Reply List: 1.OK
2. ERROR xx

Example, write the data to sector addressed by the AID 0x0801 at block 0 using key A found at index position 01, without checksum

```
!1,MW,0x0801,00,A,01,0x08010000000000000000000000000000
$0,OK,0x46
```

Example, write the data to sector addressed by the AID 0x1003 at block 0 using key A found at index position 01, with checksum

```
$1,MW,0x1003,00,A,01,0x10030000000000000000000000000000,0x47
$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 4.1 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 ≤ ss ≤ 15 for MIFARE® 1k, 2k and 4k cards
range 16 ≤ ss ≤ 31 for MIFARE® 2k and 4k cards
range 31 ≤ 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® Std 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. 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
```

Read VALUE² AID Sector Block
MV,0xaaaa,bb,k,ii

Function. Reads 4 bytes of data in the MIFARE® VALUE² format from the sector block addressed by the AID 0xaaaa and the block bb, follows MAD rules (MAD1 and MAD2 are supported), uses the given key type k found at the key index ii

where 0xaaaa is the 2 byte AID in hexadecimal

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 bb is the block number in decimal
range $00 \leq bb \leq 02$ for MIFARE® 1k, 2k 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

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

Example, read the VALUE from sector addressed by the AID 0x0801 at block 1 using key A found at index position 01, without checksum

```
!1,MV,0x0801,01,A,01
$0,V,03,01,0x00001000,0x73
```

Example, read the VALUE from sector addressed by the AID 0x1003 at block 1 using key A found at index position 01, with checksum

```
$1,MV,0x1003,01,A,01,0x6F
$0,V,03,01,0x00001000,0x79
```

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 $00 \leq ss \leq 15$ 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
```

Decrement VALUE² AID Sector Block
MD,0xaaaa,bb,k,ii,0xhhhhhhhh

Function. The 4 bytes of data will be subtracted from the VALUE² found at the sector block addressed by the AID 0xaaaa and the block bb, follows MAD rules (MAD1 and MAD2 are supported), uses the given key type k found at the key index ii

where 0xaaaa is the 2 byte AID in hexadecimal

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 0x00000002 the VALUE sector addressed by the AID 0x0801 at block 1 using key A found at index position 01, without checksum

```
!1,MD,0x0801,01,A,01,0x00000002
$0,OK,0x46
```

Example, decrement by the amount 0x00000002 the VALUE sector addressed by the AID 0x1003 at block 1 using key A found at index position 01, with checksum

```
$1,MD,0x1003,01,A,01,0x00000002,0xB3
$0,OK,0x46
```

Increment VALUE² Sector Block
A,ss,bb,k,ii,0xhhhhhhhh

Function. The 4 bytes of hex data will be added to 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 $00 \leq ss \leq 15$ 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 $00 \leq bb \leq 03$ 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, 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
```

Increment VALUE² AID Sector Block
MA,0xaaaa,bb,k,ii,0xhhhhhhhh

Function. The 4 bytes of data will be added to the VALUE² found at the sector block addressed by the AID 0xaaaa and the block bb, follows MAD rules (MAD1 and MAD2 are supported), uses the given key type k found at the key index ii

where 0xaaaa is the 2 byte AID in hexadecimal

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 ≤ 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 0x00000002 the VALUE sector addressed by the AID 0x0801 at block 1 using key A found at index position 01, without checksum

```
!1,MA,0x0801,01,A,01,0x00000002
$0,OK,0x46
```

Example, increment by the amount 0x00000002 the VALUE sector addressed by the AID 0x1003 at block 1 using key A found at index position 01, with checksum

```
$1,MA,0x1003,01,A,01,0x00000002,0xB0
$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 ≤ 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

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.

Write VALUE² AID Sector Block
MX,0xaaaa,bb,k,ii,0xhhhhhhhh

Function. Writes 4 bytes of data in the MIFARE® VALUE² format to the sector block addressed by the AID 0xaaaa and the block bb, follows MAD rules (MAD1 and MAD2 are supported), uses the given key type k found at the key index ii

where 0xaaaa is the 2 byte AID in hexadecimal

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 0x00001000 as the VALUE at the sector addressed by the AID 0x0801 block 0 using key A found at index position 01, without checksum

```
!1,MX,0x0801,00,A,01,0x00001000
$0,OK,0x46
```

Example, write the amount 0x00001000 as the VALUE at the sector addressed by the AID 0x1003 block 1 using key A found at index position 01, with checksum

```
$1,MX,0x1003,01,A,01,0x00001000,0xC6
$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.

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

Appendix A
Mifare Plus key addresses
The first four keys are mandatory for personalisation

i. Card Configuration key:	0x9001
ii. Card master key:	0x9000
iii. Level 2 switch key:	0x9002
iv. Level 3 switch key:	0x9003

in addition other optional keys are,

v. SL1 card authentication key:	0x9004
vii. MFP Configuration block:	0xB000
viii. IID:	0xB001
ix. ATS info:	0xB002
x. Field configuration block:	0xB003
xi. Select VC key:	0xA000
xii. VC polling ENC key:	0xA001
Xiii. VC polling MAC key:	0xA080
Xiv. Proximity check key:	0xA081
xv. AES sector keys:	0x4000-0x404F

Appendix B
Example to personalise a Mifare Plus card to SL1 level
- initial state SL0, keys have been arbitarily selected, choose your own, commands in blue are for information purposes only and are not required for personalisation, the commands in green are neede for pesonalisation

```
!1,PT
$0,0x18,0xBD
!1,PW,0x9000,0xAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
$0,OK,0x46
!1,PW,0x9001,0xBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
$0,OK,0x46
!1,PW,0x9002,0x22222222222222222222222222222222
$0,OK,0x46
!1,PW,0x9003,0x33333333333333333333333333333333
$0,OK,0x46
!1,PW,0x9004,0xCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
$0,OK,0x46
!1,PW,0x4000,0x00000000000000000000000000000001
$0,OK,0x46
!1,PW,0x4001,0x00000000000000000000000000000010
$0,OK,0x46
!1,PW,0x4002,0x00000000000000000000000000000002
$0,OK,0x46
!1,PW,0x4003,0x00000000000000000000000000000020
$0,OK,0x46
!1,PC
$0,OK,0x46
```
- now SL1 state