OVERVIEW

The NFC RFID reader for the Raspberry Pi is a complete read and write solution for the most common and widely used 13.56 MHz NFC cards and tags.

CognIoT continues the theme of offering a great OOB (out of box) experiences for Pi based RFID readers. Just plug the NFC reader into your Pi and it will be reading tags even before you have run any software on the Pi. Add software (example code provided) to turn your Pi unto a powerful NFC reader and interact with tags or NFC enabled phones.

The NFC RFID Reader is Pi-command compatible with the other CognIoT RFID readers allowing easy migration between the different RFID technologies.

The specific NFC RFID tag types supported are shown below:

- NFC Type 1: Topaz (Broadcom) low-cost label tags – NOT supported
- **NFC Type 2: Ultralight, MIFARE, NTAG2 – most common and widely used type in the market – FULLY SUPPORTED**
- NFC Type 3: Sony Felica – Secure Payment card used mainly in Japan and Asia – NOT SUPPORTED
- **NFC Type 4: DESfire, Smart MX, JCOP – Secure Payment card – SUPPORTED FOR SERIAL NUMBER (UID) ACQUISITION.**

To counter the relatively "open" memory structures of some of the NFC Type 2 cards, the NFC RFID Reader also includes a built-in and programmable "Dynamic" software Encryption system. This allows data to be stored on the cards in a scrambled format. The Encryption algorithm uses both the unique card serial number and user programmable "seeds" to encrypt the stored data, so the same information stored on different cards would always appear as different encrypted data on each card.

CONTENTS

CONFIGURATION

The Reader has some options to enable the user to select which GPIO of the Raspberry Pi is used.

	Jumper 1. GPIO18
	Jumper 2, GPIO17
	Jumper 3, GPIO21
	Jumper 4, GPIO22

Host must Transmit CMD/DATA within 10mS of Command Strobe going low

Only one jumper should be connected at any time. For example, connecting the jumper across the jumper 1 pins enables the Pi to select the Reader using GPIO18.

NOTE that the (programmable) polling delay period is skipped if the EEPROM parameter is zero or if there are commands to be processed.

NFC RFID READER OPERATION

The most common and widely used NFC (Type 2) cards and tags are based on ISO14443A Mifare "Classic" technology including Mifare1k / 4k, Ultralight and the NTAG2 family of cards and tags. The NFC Type 2 transponders are available in a wide range of memory sizes from 64-bytes for the simple NTAG2 / Ultralight tags to 4k (4096) bytes for Mifare4K. The 13.56 MHz carrier frequency provides fast transaction times of 106 kbaud and additional features such as security vary across the product range. The smaller memory tags tend to have "open" memory structures and the larger memory tags have advanced hardware encryption and Password (Keycode) protection for memory sectors. To increase data security across the product range, the NFC RFID Reader includes a built-in "Dynamic" software encryption system that uses the cards unique serial number as a

"seed" so the same data stored on multiple cards would always appear as different scrambled data. This feature can be turned on/off using an EEPROM parameter and can be used across all the NFC Type 2 tag types. This means Mifare1k/4k cards and tags that already have the hardware CRYPTO1 encryption and Password (Keycode) locking system can benefit from an additional level of software-encryption security as well.

Communication with the NFC Type 2 cards and tags can only proceed after mutual authentication between the NFC RFID Reader and the card has succeeded (as defined by ISO 14443A standard and handled internally by the NFC RFID Reader). During this initial authentication, the card/tag serial number (UID) is read and can be accessed using the CARD UID command (ASCII "U", 0x55) or the Automatic serial number output feature. Depending on the card/tag type, user memory may be protected by the Password (keycode) and hardware (CRYPTO1) system or the proprietary IB technology software-encryption system. If the appropriate Passwords and encryption seeds are known then the user memory can be accessed for reading and writing data.

The NFC RFID Reader is a proximity system and a Read/Write range of up to 8cm can be achieved under ideal conditions using the appropriate antenna. When power is first applied to the board the red and green LEDs flash once to indicate successful power-up (both LEDs stay on if initialisation fails). The NFC RFID Reader can also check for antenna faults and internal error conditions, these problems are indicated by the red LED or both LEDs flashing continuously until the fault has been rectified.

The NFC RFID Reader will normally have the red LED lit until a valid tag is brought into the RF field. If the tag is accepted as valid then the green LED is turned ON (and red LED OFF).

The NFC RFID Reader supports a simple command/reply protocol or an "Automatic" auxiliary output mode. In command mode, the Command Strobe signal is used for "hardware-handshaking" and when Command Strobe goes LOW, a command can be sent to the NFC RFID Reader module. The NFC RFID Reader then replies with a status/acknowledge byte followed by data as required.

For example, in command mode:

To read the card serial number (UID), when Data Strobe signal goes low send the CARD UID command (ASCII "U", 0x55).

The NFC RFID Reader will reply with 0x80 (80 hex) status/acknowledge byte if no card present

Or

0xA6 UU UU UU UU UU UU UU where 0xA6 is the status/acknowledge byte indicating valid NTAG2/UL type card and UU – UU is the 7-byte serial number.

If "Automatic" auxiliary output mode is selected (to read UID for example).

(EEPROM parameter 1 = 03 for 9600 baud serial output, EEPROM parameter 7 = 00 for UID and EEPROM parameter 8 = 01 for output from the Tx pin).

When a valid card first enters the RF field, the NFC RFID Reader will automatically output the 7-byte serial number from the Tx pin without any command being sent.

In "Automatic" auxiliary output mode, the EEPROM parameters can also be configured to automatically perform a block read and output 16-bytes of data in multiple formats. Note that if **"Automatic" auxiliary output mode** is selected and redirected to the Tx pin then commands such as PROGRAM-EEPROM can still be sent be sent and processed (when Command Strobe goes LOW) but there will be **no status/acknowledge byte reply** (so there is no conflict of data).

SUPPORTED TRANSPONDER TYPES

The NFC RFID Reader is designed to communicate with the following passive NFC type 2 transponder devices. Note that these cards and tags are fabricated by several companies and should fully comply with ISO14443A communication standard but memory sizes and security features will vary from type to type.

1) Mifare standard 1k card (MF1 IC S50 transponder) and equivalents.

2) Mifare standard 4k card (MF1 IC S70 transponder)

3) Mifare Ultralight card (MF0 IC U1 transponder)

4) NTAG2 family of cards and tags

5) Mifare ProX, Smart MX (JCOP) dual-interface card types are supported to allow single or double UID to be acquired and "Mifare" operations performed across the contactless interface. DESFire, Mifare PLUS supported for serial number acquisition.

The operation of the NFC RFID Reader and the NTAG2/Mifare transponders is described in more detail at the end of this document.

The identification codes used for authentication of serial numbers (in the identity code look-up table in EEPROM) and for the Wiegand data output are the least significant (first) 4-bytes of the UID (serial number) data (or optionally the least significant 4-bytes of the block data read from the card for Wiegand output).

SERIAL INTERFACE

This is a basic implementation of TTL-level RS232. The NFC RFID Reader does not support buffered interrupt driven input so it must control a Command Strobe signal to inhibit

communications from the Pi when it is fully occupied with RFID communication. This Command Strobe signal is connected to Raspberry Pi GPIO18 by default but it may be changed by the user. The Raspberry Pi software must check the Command Strobe signal and only send commands/data when it is in a LOW state. The Command Strobe signal is pulsed LOW for a 10ms period each polling cycle. The Pi must wait for this LOW signal and then send the command and data immediately

The Command Strobe line remains in a LOW state while the command and data bytes are being received. After the last byte of data the Command Strobe signal "times out" for 10ms and returns HIGH.

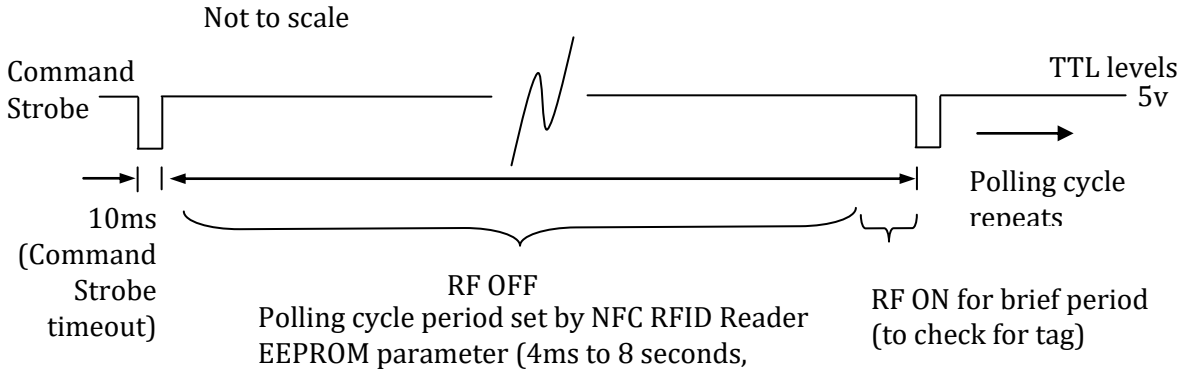
This 10ms "window" every polling cycle allows the Pi to send a single command and associated data to the NFC RFID Reader. Please note that only one command and it's corresponding data bytes can be sent during a Command Strobe LOW period, the command and data bytes must be sent with no gaps between, if there is a pause of more than 10ms between bytes then "time out" occurs, the Command Strobe line returns high and the command fails (flagged as RS232 error). The Command Strobe signal idles in this HIGH state (to inhibit Pi communication) until the next polling cycle begins.

The communication baud rate is 9600 baud, 8 bits, 1 stop, no parity.

THE NFC RFID Reader HAS THREE POLLING STATES:

1) NO CARD PRESENT AND NO PI COMMANDS RECEIVED.

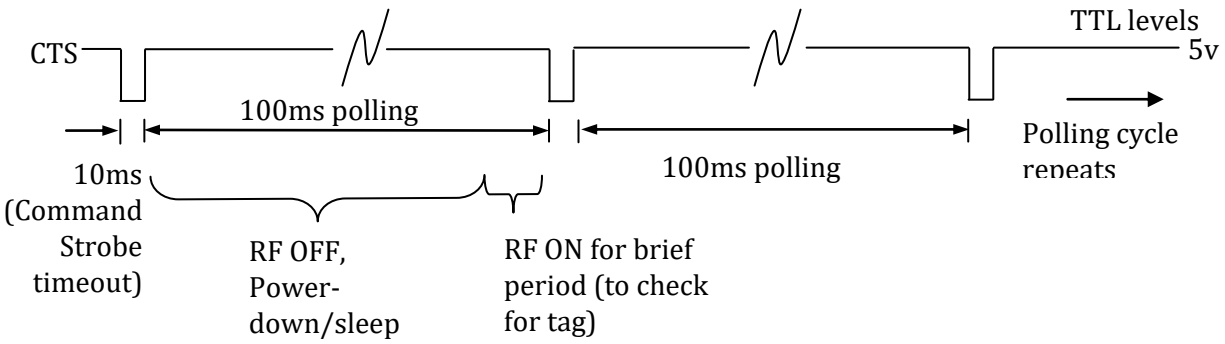
Polling cycle rate (time between subsequent Command Strobe low periods) is determined by the "polling rate" parameter stored in the NFC RFID Reader EEPROM memory. This is typically set to a long period (4ms to 8 seconds, default setting 260mS).



2) NFC CARD/TAG IN FIELD, NO PI COMMANDS RECEIVED.

When a card is detected in the field the polling rate changes to approximately 100ms (between Command Strobe low periods). This speeding up of the polling rate is to ensure that the NFC RFID Reader can respond quickly to the card leaving and a new card entering the field.

Not to scale

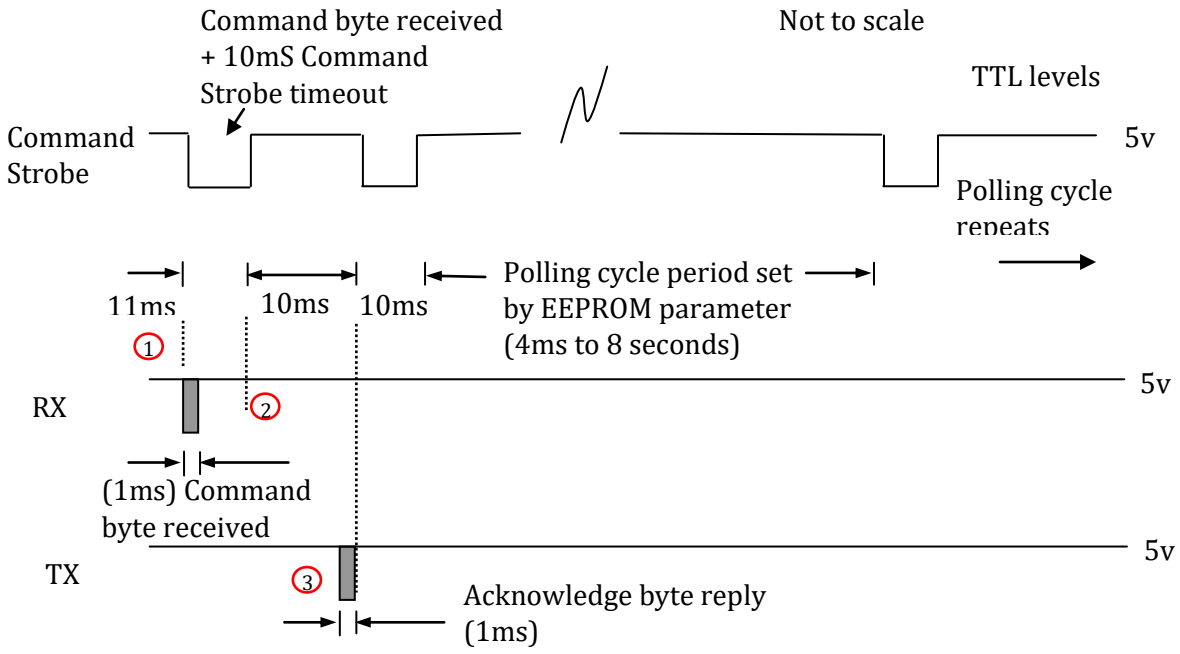


3) HOST COMMANDS RECEIVED AND PROCESSED.

When the NFC RFID Reader receives commands from the Pi computer, the polling delay is skipped to allow a quick response to the command. The NFC RFID Reader will therefore respond to the Pi command immediately after the RF communication is complete. This means that commands such as READ or WRITE BLOCK can be repeated quickly and large amounts of data handled efficiently.

Example a) NO card present, single CARD UID (0x55) command received.

Note: at 9600 baud serial communication rate, a single byte is received or transmitted in approximately 1mS (104µS per bit). If no commands follow then the polling rate reverts back to the stored parameter value as in (1).

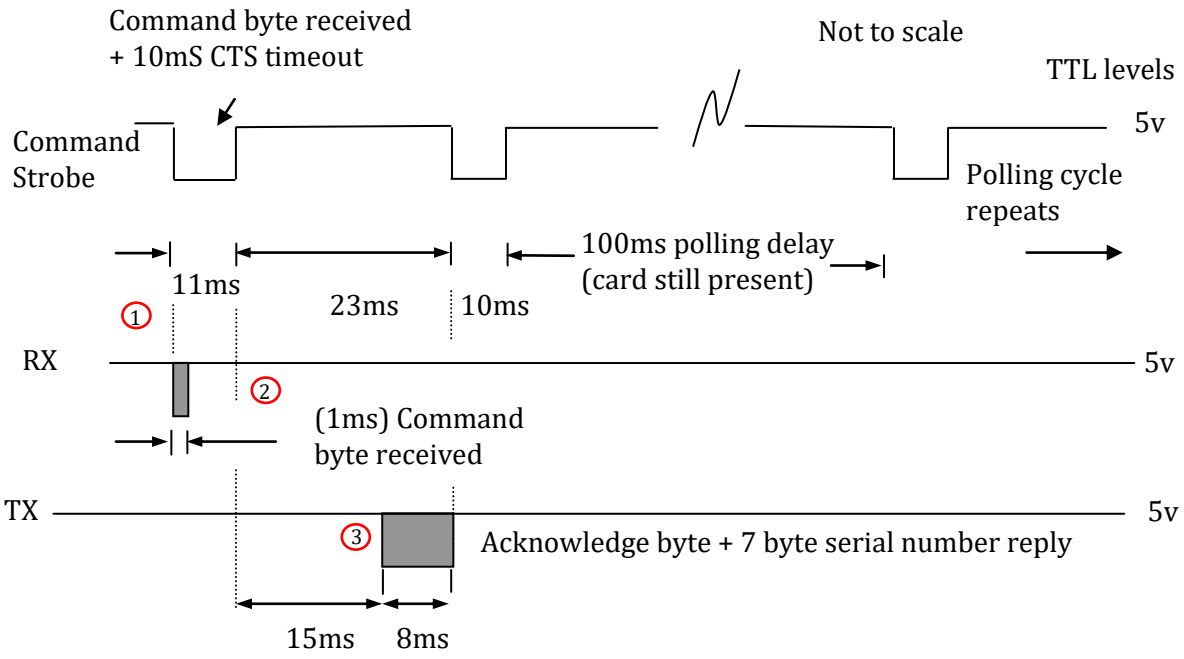


① Host waits for CTS falling edge then sends command byte.

② NFC RFID Reader processes command, RF turned ON for brief period to check if card present.

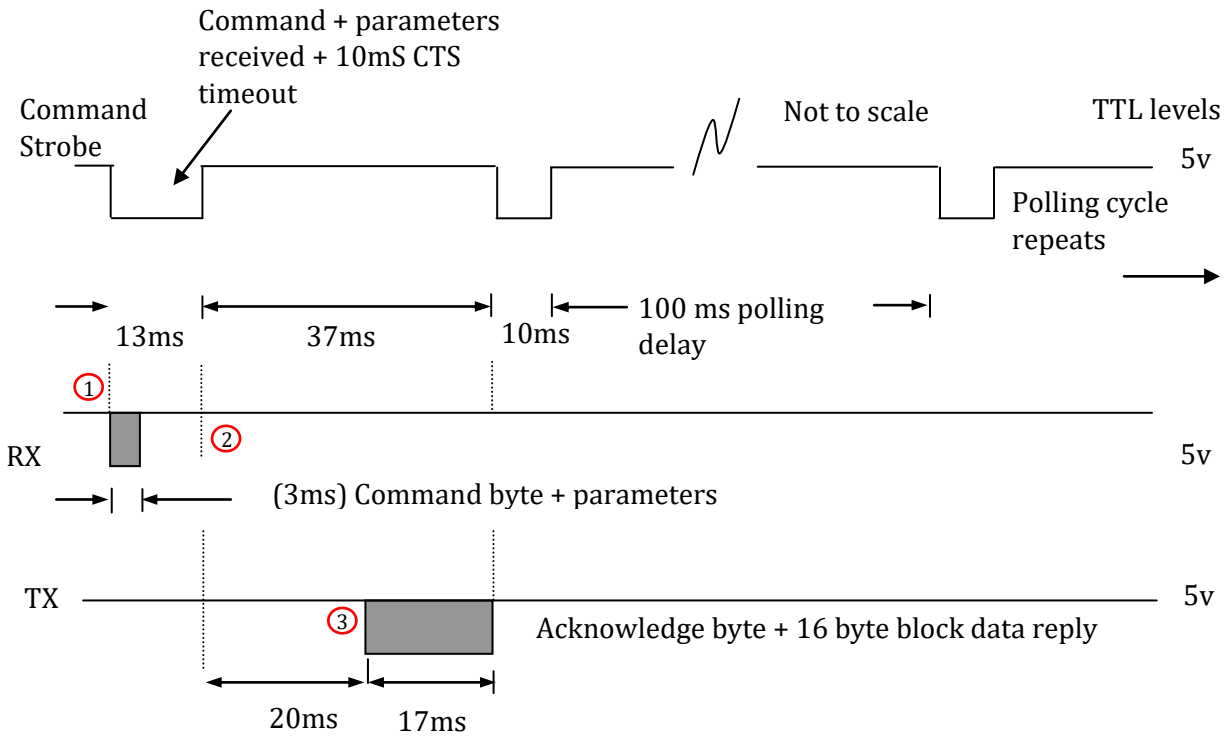
③ NFC RFID Reader then replies with acknowledge byte (+ data).

Example b) NFC card in field, single CARD UID (0x55) command received.



Example c) NFC card in field, valid READ BLOCK command received

(Read cmd (0x52) + Keycode number + Block number)



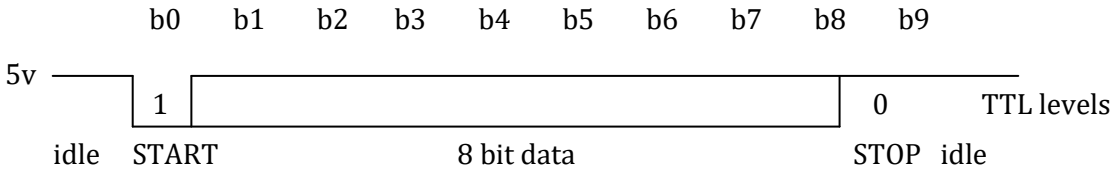
SUMMARY OF POLLING RATES AND COMMAND TIMING

Three polling rates:

1) NO card and NO commands: Polling rate determined by Polling rate parameter in NFC RFID Reader EEPROM (4mS to 8 seconds, default setting 260mS)
2) Card present but NO commands: 100ms polling delay between Command Strobe pulses.
3) Command (and parameters) received: polling delay skipped (approx. 10mS to next Command Strobe pulse).

Pi communication software must be able to handle the three polling rates.

Transmitted or Received data byte, 9600 baud, 8 bit, 1 stop, No parity (104 μS per bit)



HOST DRIVER SOFTWARE

Communication with the NFC RFID Reader module is via the TTL level RS232 interface (9600 baud, 8 bit, 1 stop bit, no parity) and uses the Command Strobe line for hardware handshaking. The Windows applications (supplied with the Evaluation kit) can be used to communicate with the module or the user can write their own application on a PC or a microcontroller. Please note that the Pi software must be able to handle the three distinct polling rates (different periods between Command Strobe pulses). The following basic communication algorithm can be used:-

TYPICAL PI COMPUTER "PSEUDO" DRIVER CODE

```
if (Green LED ON (pin 2 = 0))                // Optional check for valid tag in field
{
  if (Command Strobe = 0)                    // Wait for Command Strobe = 0 (NFC RFID Reader ready
to receive command / data)
  {
        // Command Strobe times out after 10ms so command and all parameters must be
        sent with no-
        // gaps otherwise Command Strobe times out and goes HIGH.
        // For example, send READ BLOCK 1 using KEY 0 as KEYA (0x52 0x01 0x00)

        SEND_CMD( );                         // Sent command + parameters to NFC RFID Reader

        // NFC RFID Reader sets Command Strobe = 1 after last parameter received. NFC
      RFID Reader module processes
       // command, turns on RF for short period, then sends reply.

        GET_REPLY( );                        // Get Acknowledge byte + data
       // Response to READ command is 0x80 (no tag) or 0x86 (Mifare card) + sixteen
        // bytes of DATA.
  }
}
```

COMMAND PROTOCOL

The following commands are supported. The corresponding acknowledge code should be read back by the Pi and decoded to confirm that the command was received and handled correctly. The serial bit protocol is 9600 baud, 8 bits, 1 stop, no parity (lsb transmitted first).

The status flags returned in the Acknowledge byte are as follows:

b7 b6 b5 b4 b3 b2 b1 b0

1 1 1 1 1 1 1 1

| | | | | | EEPROM error (Internal EEPROM write error)

| | | | | Card OK (Card serial number matched to identity code list)

| | | | Rx OK (Card communication and acknowledgement OK)

| | | RS232 error (Host serial communication error)

| | MF type (0 = MF 1k byte card, 1 = MF 4k byte card)

| UL type (0 = MF standard 1k/4k card, **SINGLE UID**), 1 = MF Ultralight/NTAG2 card, **DOUBLE UID**)

MFRC error (Internal or antenna fault)

Note that bit 7 is fixed so that using a Mifare 1k card, the NFC RFID Reader acknowledge response to a valid Pi command would generally be 86 (Hex), indicating that a matched (or authorised) MF 1k card is present. The MF Ultralight/NTAG2 card has a different memory structure to the standard 1k/4k MF cards so bits 4 and 5 have to be checked to determine which card type is present. Note also that only the relevant flags are set after each command as indicated in the following specification.

CARD UID

Command to return card status and UID (Unique Identifier or Serial number).

The acknowledge byte flags indicate general Mifare/NTAG2 card status.

	B7 B0	
Command:	0 1 0 1 0 1 0 1	(Ascii “U”, 0x55)

Acknowledge: 1 F F F F F F X (F = Status flags)

Data only follows if card was selected OK with no errors detected.

Reply1: D D D D D D D D (D = LS Byte of UID/Serial number from card)

Reply2: D D D D D D D D

Reply3: D D D D D D D D

Reply4: D D D D D D D D

Reply5: D D D D D D D D

Reply6: D D D D D D D D } Dummy bytes (0x00) for Mifare 1k/4k card types

Reply7: D D D D D D D D

Note that Mifare 1k and 4k cards have a four-byte serial number but Ultralight/NTAG2 cards have a seven-byte serial number. To accommodate all card types, the Card UID command returns a seven-byte field with the last three bytes padded out with 0x00 dummy bytes in the case of Mifare 1k/4k cards.

CARD STATUS

Command to return card status.

The acknowledge byte flags indicate general Mifare/NTAG2 card status.

B7 B0

Command: 0 1 0 1 0 0 1 1 (Ascii “S”, 0x53)

Acknowledge: 1 F F F F F F X (F = Status flags)

PROGRAM EEPROM

The Micro NFC RFID Reader has some internal EEPROM for storing system parameters such as polling rate and authorised identity codes (serial numbers). This command sequence allows individual bytes of the EEPROM to be programmed with new data. The data is internally read back after programming to verify successful operation. Note that due to the fundamental nature of these system parameters, incorrect data may render the system temporarily inoperable.

	B7 B0	
Command:	0 1 0 1 0 0 0 0	(Ascii "P", 0x50)
Argument1:	N N N N N N N N	(N = EEPROM memory location 0 - 255)
Argument2:	D D D D D D D D	(D = data to write to EEPROM)
Acknowledge:	1 X X X F X X F	(F = Status flags)

INTERNAL EEPROM MEMORY MAP

Polling delay parameter values (EEPROM location 0):

Parameter 0 value	Polling Delay SLEEP Period
0x00	0 mS
0x10	8 mS
0x20	16 mS
0x30	32 mS
0x40	65 mS
0x50	132 mS
0x60	**262 mS**
0x70	524 mS
0x80	1 second

(SLEEP and power-down is skipped)

0x90	2 seconds
0xA0	4 seconds
0xB0	8 seconds

Polling delay can be set from 0 to 8 seconds to give complete control over current consumption and battery life. Note that setting Polling delay = 0x00 skips the SLEEP and power-down operation so polling is as fast as possible (and current consumption is highest).

Byte 0: Polling Delay (SLEEP / Power down) period (default = 0x60 = approx 260 milliseconds)

Byte 1: Aux data output: 0x00 = OFF (NO output from OP0 / OP1),

0x01 = 24 (26) bit, Wiegand on OP0 / OP1.

0x02 = 32 (34) bit, Wiegand on OP0 / OP1.

0x03 = 9600 baud serial from OP0 / Tx (default 0x03, serial output from OP0, see parameter 8)

Byte 2: Reserved (Checksum)

Byte 3: Wiegand parity option, NO Parity = 0x00 (default)

0x01 = Even/Odd parity added

Byte 4: Aux block address on card (Mifare/NTAG2 card block address 0 – 255), default 0x01, block 1

(only used if parameter byte 8 is set to 0x01 for internal Block Read)

Byte 5: Key number / type used for internal Block Read of Aux data (not required for UL / NTAG2 type):

(TxxKKKKK), (T = Key type, 0 = KeyA, 1 = KeyB)

(K = Key code number, 0 - 31), default = key 0x00 used as typeA

(only used if parameter byte 8 is set to 0x01 for internal Block Read)

Byte 6: “Beep” delay parameter (x 40 mS) default = 0x00 (OFF)

Byte 7: Aux output source data selection.

0x00 = use UID / serial number (default)

0x01 = perform Block Read

Byte 8: Aux out (serial data) redirection (OP0 - pin 20 or Tx – pin 23)

0x00 = Serial aux output from OP0 pin (default)

0x01 = Serial aux output from main Tx pin

Byte 9: Aux output serial format (Hex or ASCII), HEX output = 0x00 (default)

ASCII output = 0x01

Byte 10: Aux output byte order option, plain data as read from card = 0x00 (default)

Byte order reversed = 0x01

Byte 11: Encryption ON/OFF control (default 00 = OFF, non-zero = ON)

Byte 12:) 32 bit Encryption Seed (M.S byte)

Byte 13:)

Byte 14:)

Byte 15:) (L.S byte)

Start of authorised card codes. List is terminated with FF FF FF FF sequence.

List is regarded as empty (all identity codes valid) if first code sequence in list is (FF FF FF FF).

List can hold up to 60 identity codes (serial numbers)

Byte 16: 0xFF Empty list

Byte 17: 0xFF

Byte 18: 0xFF

Byte 19: 0xFF

Byte 20: (MSB) Tag identity code

Byte 21:

Byte 22:

Byte 23: (LSB)

-

-

Byte 255: Last Internal EEPROM location

Note that the polling delay parameter must be a valid value (as shown in the table above), other values will give undefined results.

Default NFC RFID Reader EEPROM parameter settings:

Byte 0: 0x60, 260mS Polling delay / SLEEP period

Byte 1: 0x03, Aux data output as 9600 baud serial on OP0

Byte 2: Reserved

Byte 3: 0x00 Wiegand NO parity option (only used if Byte 1 = 0x01 / 02)

Byte 4: 0x01 Aux block address on card (only used if Byte 8 = 0x01)

Byte 5: 0x00 Key number / type used for internal Block Read of Aux data (Use Key Code 0 as Key Type A, only used if Byte 8 = 0x01)

Byte 6: 0x00 "Beep" output delay OFF

Byte 7: 0x00 Aux output source data is UID (serial number).

Byte 8: 0x00 Aux output (serial data) directed to OP0 pin.

Byte 9: 0x00 Aux output serial format, HEX byte format

Byte 10: 0x00 Aux data byte order, plain as read from card

Byte 11: 0x00 Encryption control OFF

Byte 12: 0x01) 32 bit Encryption Seed (M.S byte)

Byte 13: 0x03)

Byte 14: 0x09)

Byte 15: 0x07) (L.S byte)

STORE KEYS

The Micro NFC RFID Reader has additional internal storage for 32 x Mifare (CRYPTO1) Security KEYs. Six byte Key codes are required to access individual card sectors for any Read or Write operations on Mifare1k or 4k cards. This command sequence allows 6 byte Key codes to be stored at any one of the 32 key code locations. Factory defaults are Philips/NXP specified transport key code pairs (Hex FF FF FF FF FF FF / Hex FF FF FF FF FF FF)) and (Hex A0 A1 A2 A3 A4 A5 / Hex B0 B1 B2 B3 B4 B5) and these are stored in the NFC RFID Reader non-volatile memory on factory-reset. Note that due to the fundamental nature of these Key codes, incorrect values may render the system inoperable. Only one or two Security key codes are required to unlock a card sector so the provision of 32 storage locations allows for many possible applications and card uses.

IT IS STRONGLY ADVISED THAT THE KEY CODES IN THE NFC RFID Reader AND STORED ON THE MIFARE CARD ARE NOT CHANGED UNTIL THE OPERATION OF THE MIFARE CARD SECURITY IS FULLY UNDERSTOOD.

	B7 B0	
Command:	0 1 0 0 1 0 1 1	(Ascii “K”, 0x4B)
Argument1:	x x x K K K K K	(K = Key code number, 0 - 31)
Argument2:	D D D D D D D D	(D = data to write to EEPROM, LS byte)
Argument3:	D D D D D D D D	
Argument4:	D D D D D D D D	
Argument5:	D D D D D D D D	
Argument6:	D D D D D D D D	
Argument7:	D D D D D D D D	(D = data to write to EEPROM, MS byte)

Acknowledge: 1 X X X F X X F (F = Status flags)

INTERNAL KEY STORAGE MEMORY MAP (DEFAULT SETTINGS)

Location 0 (0x00): Key code 0 (Default 0xFF FF FF FF FF FF)

Location 1 (0x01): Key code 1 (Default 0xFF FF FF FF FF FF)

Location 2 (0x02): Key code 2 (Default 0xA0 A1 A2 A3 A4 A5)

Location 3 (0x03): Key code 3 (Default 0xB0 B1 B2 B3 B4 B5)

-

-

-

Location 28 (0x1C): Key code 28 (Default 0xFF FF FF FF FF FF)

Location 29 (0x1D): Key code 29 (Default 0xFF FF FF FF FF FF)

Location 30 (0x1E): Key code 30 (Default 0xA0 A1 A2 A3 A4 A5)

Location 31 (0x1F): Key code 31 (Default 0xB0 B1 B2 B3 B4 B5)

Note that Mifare cards manufactured by Philips/NXP and licensed equivalents have default transport key codes of (0xFF FF FF FF FF FF) or (0xA0 A1 A2 A3 A4 A5 and 0xB0 B1 B2 B3 B4 B5). The NFC RFID Reader NFC has both pairs stored as default settings to allow ease of use when the system is first used. (More information on the Mifare card memory maps, Security Keys and the KeyA and KeyB types can be found at the end of this document).

WRITE CARD BLOCK

Command to write 16 bytes of data to specified Mifare/NTAG2 block. A Mifare Block is made up of 16 bytes and there are four blocks in each card sector (sixteen blocks per sector in upper half of Mifare 4k card). Note that blocks 3, 7, 11, 15 etc are sector trailer blocks that contain Security Key data and Access bits. Writing incorrect information to these blocks can permanently disable the sector concerned. The first argument is the block number to write data to, the second argument specifies which key code (0 - 31 from the internal storage area) to use for sector authentication/unlocking and if the Security Key is to be used as a KeyA or KeyB type code. If the

write was unsuccessful (invalid card, authentication failed or card out of field) then Status flags in acknowledge byte indicate error.

	B7 B0	
Command:	0 1 0 1 0 1 1 1	(Ascii "W", 0x57)
Argument1:	N N N N N N N N	(N = MF Card Block Address 0 – 255)
Argument2:	T x x K K K K K	(T = Key Type, 0 = KeyA, 1= KeyB) (K = Key code number, 0 – 31)
Argument3:	D D D D D D D D	(D = LS Byte of data to write to card)
Argument4:	D D D D D D D D	
Argument5:	D D D D D D D D	
Argument6:	D D D D D D D D	
↓		16 Bytes of data
Argument15:	D D D D D D D D	
Argument16:	D D D D D D D D	
Argument17:	D D D D D D D D	
Argument18:	D D D D D D D D	(D = MS Byte of data to write to card)
Acknowledge:	1 F F F F F F X	(F = Status flags)

Note that Ultralight / NTAG2 cards DO NOT USE Security Keys or CRYPTO Authentication and the memory is organised differently as groups of 4 bytes (Pages). Only one Page of 4 bytes can be written at a time so to maintain compatibility and a simple NFC RFID Reader Pi command set, the same command as above is used to write data to Ultralight/NTAG2 cards. The command and arguments have the same structure but different meanings. The "Block" address is treated as a "Page Address" and the KeyType/Key number parameter is a dummy 0x00 byte. In addition the 4 bytes of data are padded out to 16 bytes with dummy 0x00 bytes.

	B7 B0	
Command:	0 1 0 1 0 1 1 1	(Ascii "W", 0x57)

Argument1:	x x N N N N N N	(N = UL/NTAG2 Card Page Address 0 – 15/63)
Argument2:	0 0 0 0 0 0 0 0	(Dummy byte, 0x00)
Argument3:	D D D D D D D D	(D = LS Byte of data to write to UL card)
Argument4:	D D D D D D D D	
Argument5:	D D D D D D D D	
Argument6:	D D D D D D D D	(D = MS Byte of data to write to UL card)

Argument7 – Argument18

0 0 0 0 0 0 0 0 12 Dummy padding bytes, 0x00

Acknowledge: 1 F F F F F F X (F = Status flags)

READ CARD BLOCK

Command to read 16 bytes of data from specified Mifare/NTAG2 block. The first argument is the block number to read data from, the second argument specifies which key code (0 - 31 from the internal storage area) to use for sector authentication/unlocking and if the Security Key is to be used as a KeyA or KeyB type code. If the read was successful, indicated by acknowledge status flags then sixteen bytes of block data follow.

	B7 B0	
Command:	0 1 0 1 0 0 1 0	(Ascii “R”, 0x52)
Argument1:	N N N N N N N N	(N = MF Card Block Address 0 – 255)
Argument2:	T x x K K K K K	(T = Key Type, 0 = KeyA, 1= KeyB)
		(K = Key code number, 0 – 31)
Acknowledge:	1 F F F F F F X	(F = Status flags)

Data only follows if Read was successful

Reply1:	D D D D D D D D	(D = LS Byte of data Read from card)
Reply2:	D D D D D D D D	
Reply3:	D D D D D D D D	
Reply4:	D D D D D D D D	
↓		
		16 Bytes of data
Reply13:	D D D D D D D D	
Reply14:	D D D D D D D D	
Reply15:	D D D D D D D D	
Reply16:	D D D D D D D D	(D = MS Byte of data Read from card)

Note that as mentioned for the WRITE command, Ultralight/NTAG2 cards DO NOT USE Security Keys or Authentication and the memory is organised differently as groups of 4 bytes (Pages). However, unlike the Write command, 16 bytes (4 pages) can be read in a single operation. The same Read command as above is used except the "Block" address is treated as a "Page Address" and the KeyType/Key number parameter is a dummy 0x00 byte. For page numbers beyond the end of the user memory, the card data wraps around to page 0 etc.

	B7 B0	
Command:	0 1 0 1 0 0 1 0	(Ascii "R", 0x52)
Argument1:	x x N N N N N N	(N = UL/NTAG2 Card Page Address 0 – 15/63)
Argument2:	0 0 0 0 0 0 0 0	(Dummy byte, 0x00)
Acknowledge:	1 F F F F F F X	(F = Status flags)

Data only follows if Read was successful

Reply1:	D D D D D D D D	(D = LS Byte of data Read from UL card)
Reply2:	D D D D D D D D	

Reply3:	D D D D D D D D	
Reply4:	D D D D D D D D	
↓		
		16 Bytes of data
Reply13:	D D D D D D D D	
Reply14:	D D D D D D D D	
Reply15:	D D D D D D D D	
Reply16:	D D D D D D D D	(D = MS Byte of data Read from UL card)

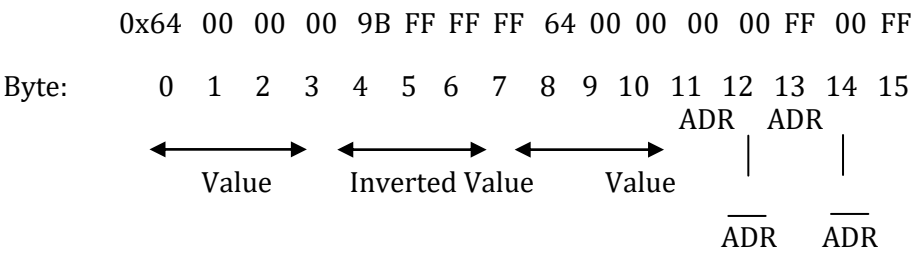
INC VALUE (ONLY OPERATES ON VALUE DATA STRUCTURE)

Command to increment integer within a “Mifare” Value Data Structure. The command loads the value from the specified block address, adds the integer parameter and stores the result at the same or another block address. Note that the source block must have been formatted as a Value Block beforehand according to the data structure below, using the WRITE command. The INC Value command only operates on a "Value Block Structure" and will fail if the block configuration or the specified key type is incorrect.

VALUE BLOCK STRUCTURE

Example format for value = 100 decimal (0x64), at block address 0.

(Value data stored LS byte first, ADR = block address, $\overline{\text{ADR}}$ = inverted block address)



The first argument is the source block address to load data from, the second argument specifies which key code and type to use for sector authentication (0-31 and if it is KeyA or KeyB type).

The third argument specifies the destination block address where the incremented data is stored. Note that source and destination blocks must be within same authenticated sector. The four byte positive integer to add follows (least significant byte first).

	B7 B0	
Command:	0 1 0 0 1 0 0 1	(Ascii "I", 0x49)
Argument1:	N N N N N N N N	(N = MF source block address 0 – 255)
Argument2:	T x x K K K K K	(T = Key Type, 0 = KeyA, 1= KeyB) (K = Key code number, 0 – 31)
Argument3:	N N N N N N N N	(N = MF destination block address 0 – 255)
Argument4:	D D D D D D D D	(D = LS byte of integer to add)
Argument5:	D D D D D D D D	4 byte integer
Argument6:	D D D D D D D D	
Argument7:	D D D D D D D D	(D = MS byte of integer to add)
Acknowledge:	1 F F F F F F X	(F = Status flags)

DEC VALUE (ONLY OPERATES ON VALUE DATA STRUCTURE)

Command to decrement integer within a "Mifare" Value Data Structure. The DEC Value command operates as the INC command except the integer parameter is subtracted from the loaded value. The first argument is the source block address to load data from, the second argument specifies which key code and type to use for sector authentication (0-31 and if it is KeyA or KeyB type). The third argument specifies the destination block address where the decremented data is stored. Note that source and destination blocks must be within same authenticated sector. The four byte positive integer to subtract follows (least significant byte first).

	B7 B0	
Command:	0 1 0 0 0 1 0 0	(Ascii "D", 0x44)

Argument1:	N N N N N N N N	(N = MF source block address 0 – 255)
Argument2:	T x x K K K K K	(T = Key Type, 0 = KeyA, 1= KeyB) (K = Key code number, 0 – 31)
Argument3:	N N N N N N N N	(N = MF destination block address 0 – 255)
Argument4:	D D D D D D D D	(D = LS byte of integer to subtract)
Argument5:	D D D D D D D D	4 byte integer
Argument6:	D D D D D D D D	
Argument7:	D D D D D D D D	(D = MS byte of integer to subtract)
Acknowledge:	1 F F F F F F X	(F = Status flags)

TRANSFER VALUE (ONLY OPERATES ON VALUE DATA STRUCTURE)

Command to transfer (copy) “Mifare” Value Data Structure. The command loads the value from the specified block address and then stores the result at the same or another block address. As with INC and DEC commands the source block must have been formatted as a Value Block beforehand and the block addresses must be within same authenticated sector.

The first argument is the source block address to load data from, the second argument specifies which key code to use for sector authentication (0-31) and if it is a KeyA or KeyB code. The third argument specifies where the data is stored.

	B7 B0	
Command:	0 1 0 1 0 1 0 0	(Ascii “T”, 0x54)
Argument1:	N N N N N N N N	(N = MF source block address 0 – 255)
Argument2:	T x x K K K K K	(T = Key Type, 0 = KeyA, 1= KeyB) (K = Key code number, 0 – 31)
Argument3:	N N N N N N N N	(N = MF destination block address 0 – 255)
Acknowledge:	1 F F F F F F X	(F = Status flags)

If the Inc, Dec or Transfer function was unsuccessful (invalid card, card out of field, authentication failed or data structures are incorrect) then Status flags in acknowledge byte indicate error. Note that the value manipulation commands operate internally on the Mifare card and no data is transferred back to the NFC RFID Reader. Note also that Ultralight/NTAG2 cards do not support Value Data Structures or the Inc, Dec, Transfer commands.

TYPE IDENTIFICATION

Command to return the **ATQA** (Answer to Request, Type A) two-byte codes and the **SAK** (Select Acknowledge) single-byte code after the complete UID has been acquired. As part of the initial communication with the Mifare card (as defined by ISO 14443A specification), the Mifare transponder responds to REQA (Request Command, Type A) with ATQA. The two-byte ATQA contains information that allows particular transponder types to be indentified. Following on from this the Mifare transponder responds to the SELECT (Select Command, Type A) with SAK (Select Acknowledge, Type A). The SAK code is a single byte value that contains further information about the type of transponder and the length of the UID. The SAK value reported is the final value after all "cascade levels" and the complete UID has been acquired.

NOTE THAT THIS COMMAND IS INCLUDED FOR DIAGNOSTIC PURPOSES TO ALLOW THE USER TO DETERMINE THE EXACT TYPE OF MIFARE CARD PRESENT IN THE FIELD, IF REQUIRED.

	B7 B0	
Command:	0 1 1 1 1 0 0 0	(Ascii "x", 0x78)
Acknowledge:	1 F F F F F F X	(F = Status flags)

Data only follows if card was selected OK with no errors detected.

Reply1:	D D D D D D D D	ATQA - MSB
Reply2:	D D D D D D D D	ATQA - LSB
Reply3:	D D D D D D D D	SAK

	MF UL	MF 1K	MF 4K	MF DESFire	MF Prox	MF Prox	MF Prox	MF Prox	MF Prox	MF Prox
ATQA - MSB	0x00	0x00	0x00	0x03	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX
ATQA - LSB	0x44	0x04	0x02	0x44	0x08	0x04	0x02	0x48	0x44	0x42

	Smart MX	Smart MX	Smart MX	Smart MX	Smart MX	Smart MX
ATQA - MSB	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX
ATQA - LSB	0x08	0x04	0x02	0x48	0x44	0x42

	MF UL	MF 1K	INFINEON 1K	MF 4K	MF DESFire	MF ProX	MF ProX	MF ProX	MF ProX	MF ProX	MF ProX
SAK	0x00	0x08	0x88	0x18	0x20	0x20	0x08	0x28	0x00	0x20	0x08

	MF ProX	MF ProX	MF ProX	Smart MX	Smart MX	Smart MX	Smart MX	Smart MX	Smart MX
SAK	0x28	0x18	0x38	0x00	0x20	0x08	0x28	0x18	0x38

Note that many of the "extended" Mifare card types are complex dual interface cards with embedded microcontrollers running "chip and pin" applications. The contactless Mifare interface in these cases is typically entirely separate to the contact based interface and processor system.

MESSAGE

Command to return product and firmware identifier string to Pi.

	B7 B0	
Command:	0 1 1 1 1 0 1 0	(Ascii "z", 0x7A)
Reply:	"m IDE NFC RFID Reader Mifare AS/WD (SECMF_CWAX_LP V1.30 DD/MM/YY) copyright: IB Technology (Eccel Technology Ltd)" 0x00	

Returned string identifies product descriptor, project name, firmware version no. and date of last software change together with IB Technology copyright message. Note that the string is always NULL terminated. The string begins with a unique lower case character that can be used to identify a particular version of Micro NFC RFID Reader.

FACTORY RESET

Command to restore Factory default EEPROM values and Stored Keys and perform hardware Reset operation. The 0x55 0xAA parameters protect against accidental operation.

After Reset, the Red LED will flash 5 times indicating the successful loading of the Factory default values.

	B7 B0	
Command:	0 1 0 0 0 1 1 0	(Ascii "F", 0x46)
Argument1:	0 1 0 1 0 1 0 1	0x55
Argument1:	1 0 1 0 1 0 1 0	0xAA

Reset occurs after the command is processed so there is no Acknowledge byte reply.

ADDITION NOTES FOR COMMANDS

NOTE also that for the “Read Card Block” or “Card UID” command, if an error flag has been set in the Acknowledge code then there will be NO following data.

NOTE that the serial communication uses hardware handshaking to inhibit the Pi from sending the Micro NFC RFID Reader commands while Card communication is in progress. The serial communication system and protocol allows for a 10ms ‘window’ every Card polling cycle indicated by the Command Strobe/BUSY line being low. During this ‘window’ the Pi must assert the first start bit and start transmitting data. The Command Strobe/BUSY goes high again 10ms after the last stop bit is received. NOTE that only one command sequence is handled at a time. The period between the Command Strobe pulses (polling delay) can have three rates depending on whether a card is present or not and if commands are being received by the NFC RFID Reader

NOTE that the commands and parameters must be sent to the NFC RFID Reader with no gaps otherwise communication timeout occurs and the NFC RFID Reader enters the polling delay period (the command string would then be incomplete and an RS232 error is flagged).

NOTE that the NFC RFID Reader NFC version performs fast polling cycles as long as there are commands to be processed. As soon as the commands stop being sent or the gap between sending commands is too long then timeout occurs and the polling delay increases to 100ms (if the card is still present) or the typically longer period (as set by EEPROM parameter) if there is no card. This is to allow repeated commands to be handled quickly such as for a “complete card read” where repeated Read Block commands are sent to the NFC RFID Reader.

Commands sent infrequently will have the full polling delay between each Command Strobe/BUSY period.

BASIC NFC RFID READER COMMUNICATION

For basic operation of the NFC RFID Reader connected to a Pi computer, the NFC RFID Reader can either be polled or communication can be triggered by an interrupt signal (Green LED output). In either case the user would initially set preferred EEPROM configuration parameters and optionally use the “STORE KEYS” command to load a custom security key into the NFC RFID Reader memory or simply use the pre-loaded default key values.

For a polling technique, the Pi computer would then keep sending the “STATUS” or “CARD UID” command and would monitor the acknowledgement code until a valid Mifare/NTAG2 card was detected.

For the interrupt technique, the Green LED output can be used as an interrupt signal connected to the Pi computer. The Green LED output is normally high and goes low only when a valid card has been detected. This falling-edge signal can trigger a Pi interrupt to then send the “STATUS” or “CARD UID” command to determine the card type and serial number.

In both cases once a valid card has been detected a “READ BLOCK” or “WRITE BLOCK” command can be sent and the acknowledge code monitored to establish that the operation was successful.

METHOD OF OPERATION

The system works on a polling principle whereby the RF field is only turned on for a short period to check if a card is present. Authentication and Read/Write operations can then be performed before the RF field is turned off again and the process repeats. The polling principle where the RF is off most of the time allows for low average power consumption.

When a Mifare/NTAG2 card is detected in the field a multi-pass handshaking procedure takes place where card information and serial number data is exchanged and checked for integrity. Once this procedure has completed successfully an individual card has been selected and is available for other operations.

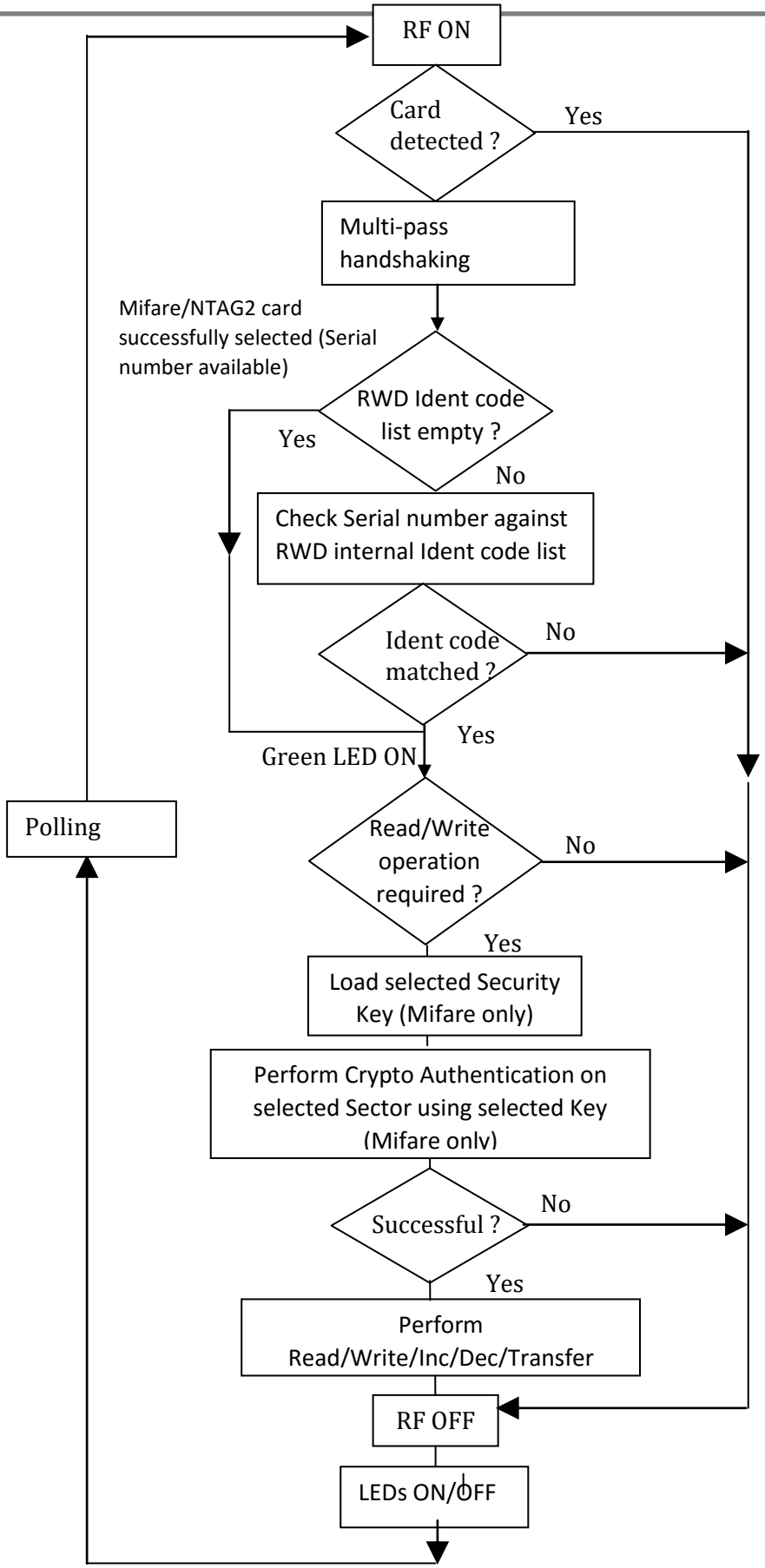
The NFC RFID Reader itself has the additional feature of then checking the four byte Mifare UID/serial number (or least significant four bytes of Ultralight/NTAG2 UID) against an internal authorisation list. The NFC RFID Reader internal EEPROM contains a list of four byte Identity codes (up to 60 of them) located from byte 12 onwards. If the list has FF FF FF FF (hex) stored at the first location (EEPROM bytes 12 - 15) then the list is treated as empty so the Identity code check is skipped.

Otherwise the card serial number is checked against all the entries in the list (until the FF FF FF FF termination code is reached) and if matched then the NFC RFID Reader allows the card to be accessed for other operations. If not the Red LED remains on and the card is blocked for further access. This is an additional level of security that can be used as a “mini access control” system for simple applications that only involve the serial number or where the Security Keys are not known.

Once the NFC RFID Reader has selected the card and has matched the serial number against its internal list (or the list is empty) then the Read/Write (or Inc/Dec/Transfer) operations can be performed. For Mifare cards these use an internal high-security Authentication Crypto algorithm that use the supplied Security Keys to gain access to a particular sector. If the Key selected does not match the Key stored in the Mifare card sector then the operation fails and the Red LED is turned on again. For Ultralight/NTAG2 cards (and as additional security on Mifare cards) the NFC RFID Reader “Dynamic” software encryption system can be used to scramble stored data.

So in summary, a card can be successfully selected but can be blocked by the NFC RFID Reader authorisation list and fail Read/Write operations because the Keys are incorrect or software encryption keys are not known.

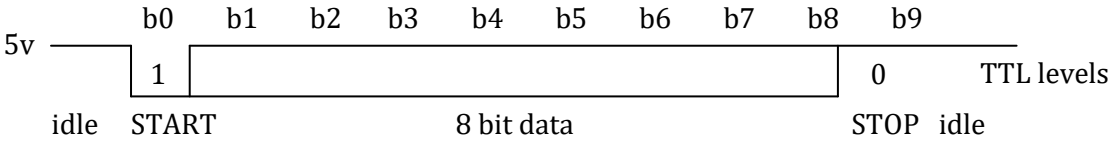
No


RF ON
Card detected ?
Yes
Multi-pass handshaking
Mifare/NTAG2 card successfully selected (Serial number available)
RWD Ident code list empty ?
Yes
No
Check Serial number against RWD internal Ident code list
Ident code matched ?
No
Yes
Green LED ON
Polling
Read/Write operation required ?
No
Yes
Load selected Security Key (Mifare only)
Perform Crypto Authentication on selected Sector using selected Key (Mifare only)
Successful ?
No
Yes
Perform Read/Write/Inc/Dec/Transfer
RF OFF
LEDs ON/OFF

AUXILIARY ASYNCHRONOUS SERIAL OUTPUT

If selected, data can be automatically output from the **OP0 or main TX** pin as 7 or 16-bytes of data transmitted asynchronously at 9600 baud, 8-bits, 1 stop-bit, no parity. The data source can be selected as the 4/7-byte UID (serial number), or the 16-bytes of automatically read Block data (1 x 16-byte Mifare block or 4 x 4-byte UL/NTAG2 Pages).

Data bytes transmitted at 9600 baud, 8-bits, 1-stop bit, No parity (104 µS per bit)



AUXILIARY WIEGAND OUTPUT PROTOCOL

If selected, data can be automatically output from the **OP0 / OP1** pins as Data HIGH and Data LOW signals according to the Wiegand protocol.

The Wiegand protocol (24 bit data length) can be made up of a leading even parity bit (for b0 - b11), 24 bits of data (from transponder data) and a trailing odd parity bit (for b12- b23) creating a 26 bit output stream. The 32-bit mode has the same format except least significant four bytes of block data are used to form the data sequence. The parity bits are included or omitted and the byte order is reversed according to the EEPROM parameter settings.

For Example:-

Mifare/NTAG2 block data (least significant 4 bytes of 16): 0x04 60 22 12

Full block data (example): 0x04 60 22 12 23 24 25 26 27 28 29 2A 2B 2C 2D 2E

(reversed byte option would use 0x2E 2D 2C 2B as base data)

Wiegand 26 bit sequence:- E (b0 --------- b11) (b12 -------- b23) O

E (0 4 6 0 2 2) O

1 0000 0100 0110 0000 0010 0010 1

Where E is EVEN parity bit for bit 0 to 11 and O is ODD parity bit for bits 12 to 23

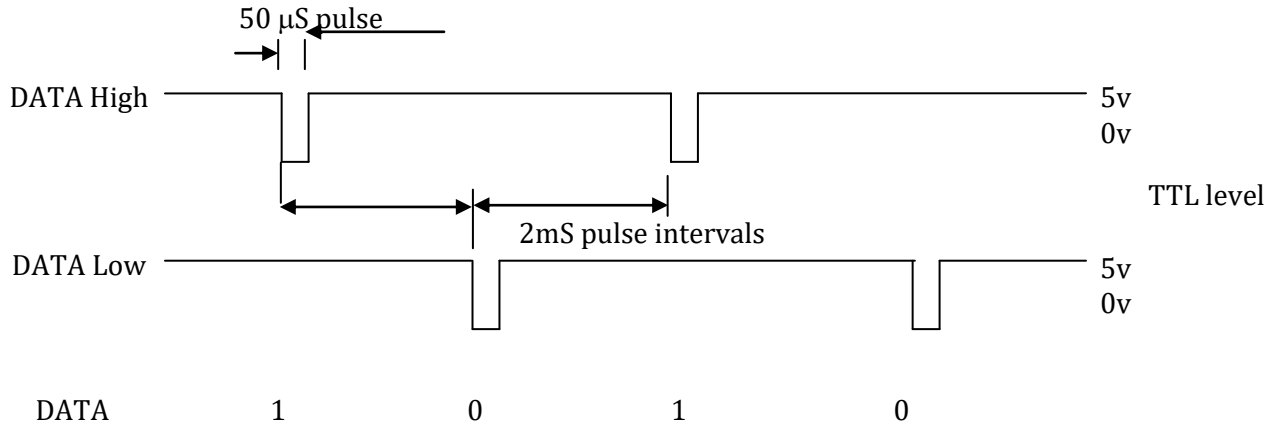
The base data for the Wiegand output can be the least significant 4-bytes of UID/serial number acquired during the initial ISO14443A communication or the least significant 4-bytes of data from a block of card memory (acquired by an internal Block Read operation). Selection is by means of an NFC RFID Reader EEPROM parameter.

For the internal Block Read operation, the block number, key code number and type (KeyA or KeyB) are programmable EEPROM parameters. In addition, parameters control whether the base data byte order is reversed or if parity bits are added before output.

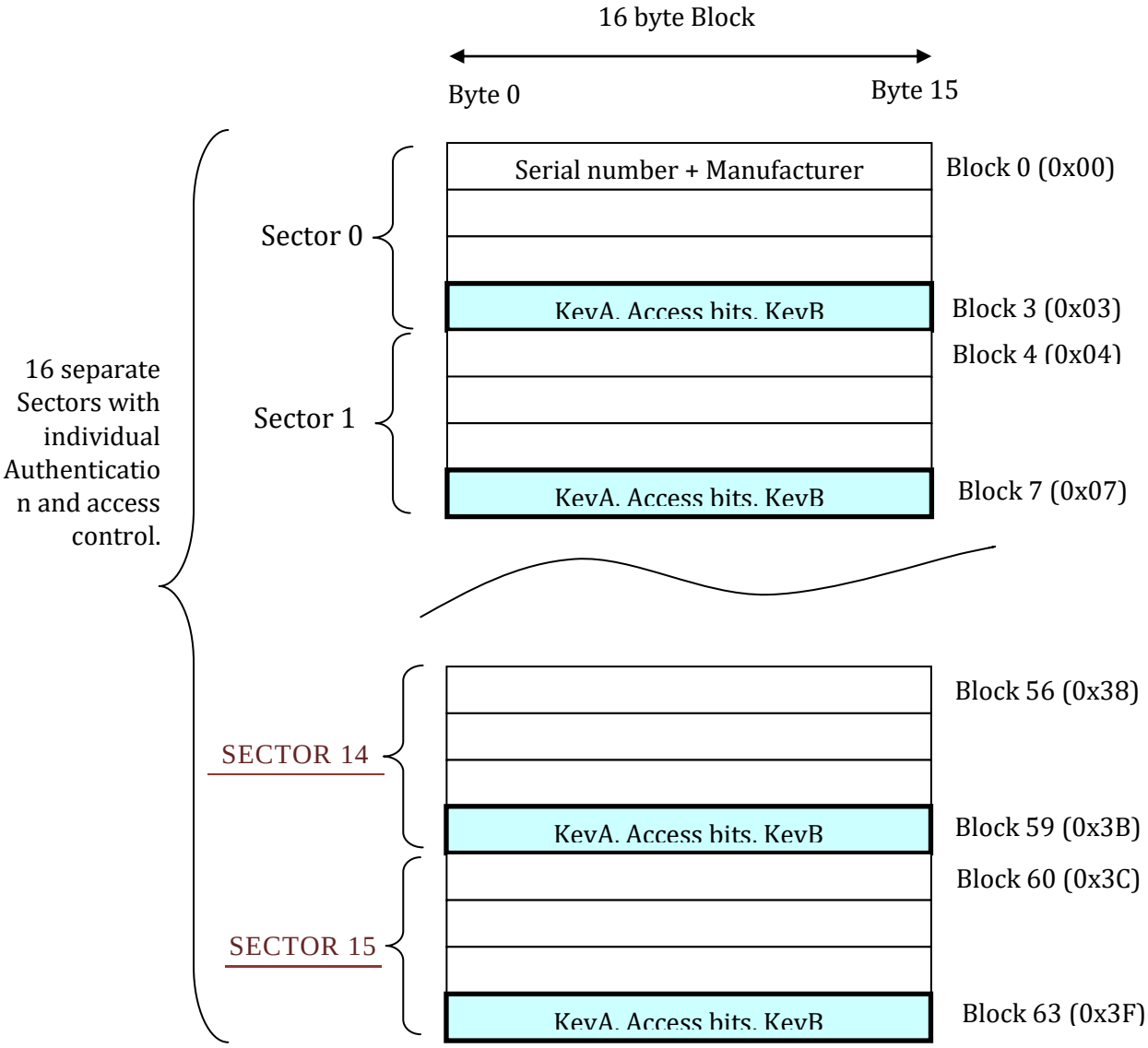
In this manner the user can select to use the UID (serial number) or user programmed information within a memory block as the base data for the Wiegand output. The complete data frame is output whenever the tag is within the NFC RFID Reader's antenna field and the tag has been validated. This output is independent of the normal TTL serial interface which responds to received commands and replies with the data as requested.

The physical Wiegand protocol is asynchronously transmitted as low going 50 µS pulses on the appropriate DATA low or DATA high pins. These pulses are separated by 2mS periods. The Wiegand sequence is output a single time whenever a valid tag enters the RF field for the first time. (NO Wiegand output if AUX OUTPUT parameter is ZERO/OFF).

WIEGAND PROTOCOL TIMING DIAGRAM



MIFARE 1K (1024 BYTE) MEMORY MAP



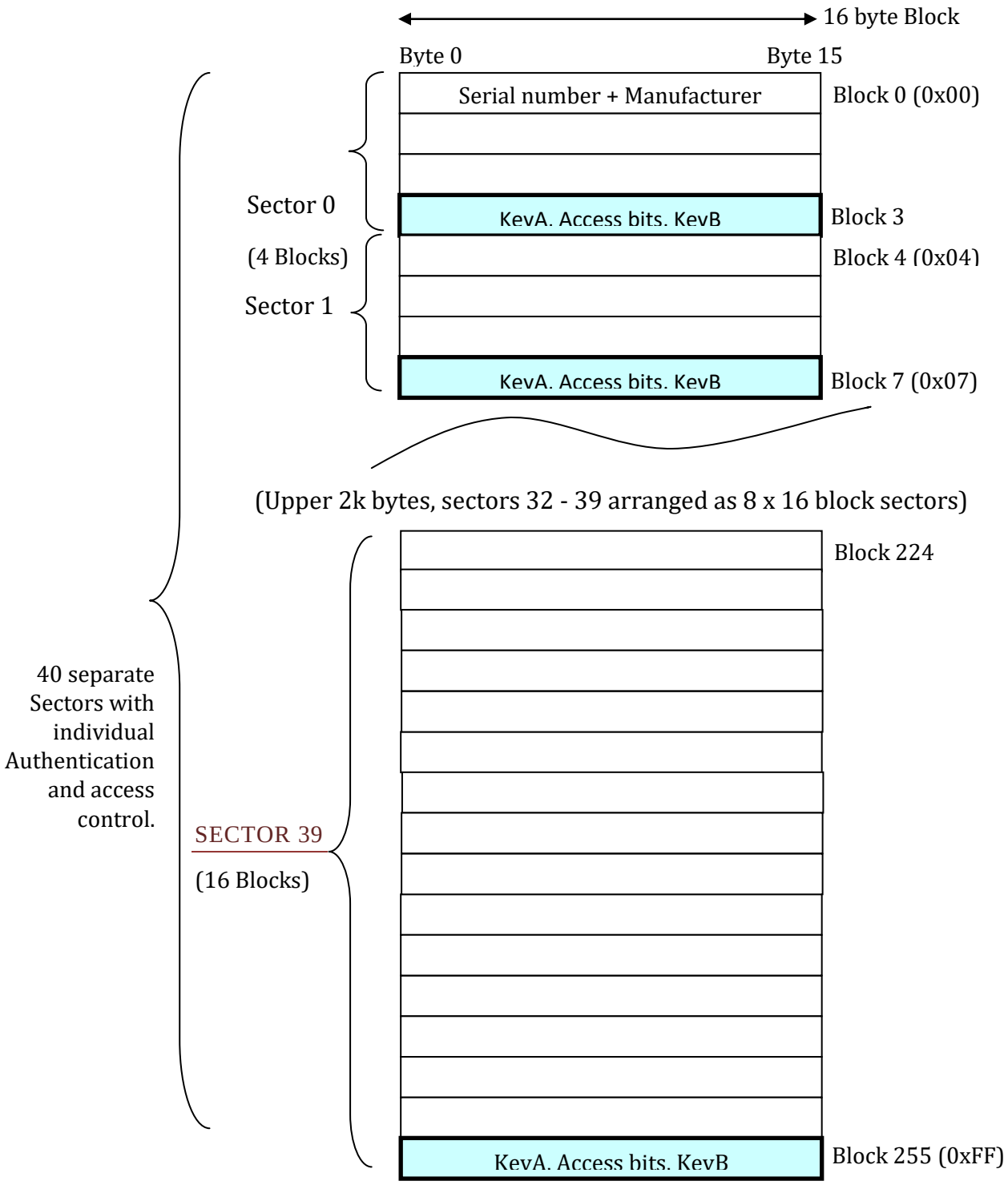
1024 byte memory is organised as sixteen sectors, each of which is made up of four blocks and each block is 16 bytes long. The first block in the memory (Block 0) is read-only and is set in the factory to contain the four-byte serial number (UID), check bytes and manufacturers data.

The last block of each sector (Blocks 3, 7, 11, 15……59, 63) is the Sector Trailer Block which contains the two security Key codes (KeyA and KeyB) and the Access bits that define how the sector can be accessed

Taking into account the Serial Number/Manufacturers Block and the Sector Trailer Blocks then there are 752 bytes of free memory for user storage. For all Read and Write operations the Mifare card memory is addressed by Block number (in hexadecimal format).

MIFARE 4K (4096 BYTE) MEMORY MAP

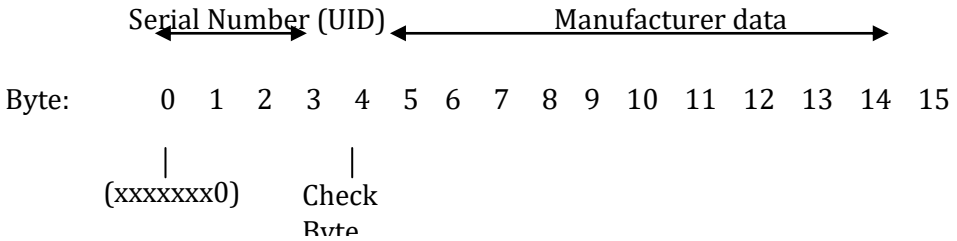
(Lower 2k bytes, sectors 0-31 arranged as 32 x 4 block sectors)



The lower 2048 bytes of the 4k card (sectors 0 – 31) are organised in the same way as the 1k card. However the upper 2048 bytes are organised as eight large sectors of 16 blocks each (sectors 32 – 39). Taking into account the Serial Number/Manufacturers Block and the Sector Trailer Blocks then there are 3440 bytes of free memory for user storage.

MANUFACTURER BLOCK

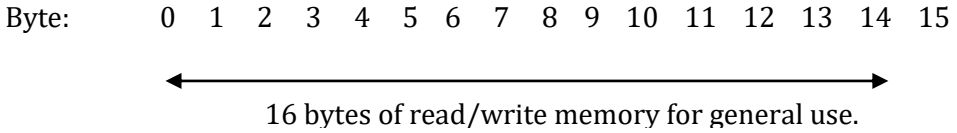
The Manufacturer block is the first data block in Sector 0 and it contains the read-only serial number (UID – Unique Identifier) and the IC manufacture information.



DATA BLOCKS

Mifare sectors contain 3 blocks (1k card) or 15 blocks (upper half of 4k card) of 16 data bytes (except Sector 0 that has Manufacturer Block and 2 data blocks). Data blocks can be configured as standard read/write memory or "Value Blocks" for special electronic-purse operations. "Value Blocks" can use additional commands such as Increment and Decrement for direct control of the data field, they have a fixed data format which permits error detection/correction and backup management. "VALUE" is a signed four byte integer (2's complement format) and is stored three times, twice non-inverted and once inverted. "ADR" signifies a one byte address that can be used to store the block number. "ADR" is stored four times, twice inverted and twice non-inverted.

Data Block

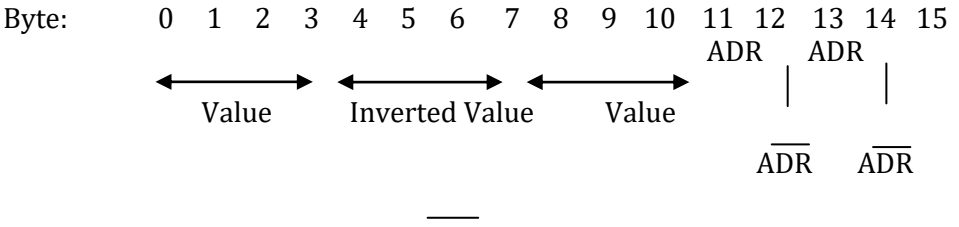


Value Block

Example format for value = 100 decimal (0x64), at block address 0.

(Value data stored LS byte first, ADR = block address, $\overline{ADR}$ = inverted block address)

0x64 00 00 00 9B FF FF FF 64 00 00 00 00 FF 00 FF



Note that the ADR and inverted ADR block address is part of the required format for the Value Data Structure BUT it is not changed by the Inc, Dec or Transfer commands and exists to allow optional storage of block numbers. For general use, the 0x00 address and 0xFF inverted address can be used to satisfy the structure format. The Inc, Dec and Transfer commands first check the structure format before beginning the operations and the commands fail if the format is not correct.

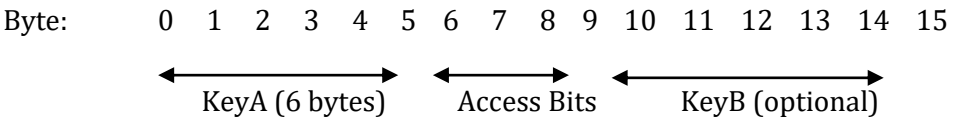
Value Data Structures must first be formatted as above using a WRITE Block command before INC, DEC or TRANSFER commands can be used.

SECTOR TRAILER BLOCK

The last block of each sector (Blocks 3, 7, 11, 15……59, 63 etc) is the Sector Trailer Block which contains the two security Key codes (KeyA and KeyB) and the Access bits that define how the data blocks can be accessed (Read/Write, Read or Write only, as data or Value blocks and using which key). If KeyB is not used then the last 6 bytes of the Sector Trailer Block can be used for general data storage. Byte 9 (last byte of Access bits) is not used and can also be used for general storage. Note that the KeyA (and KeyB) value read back as logical 0's to ensure system security.

IT IS STRONGLY RECOMMENDED THAT THE KEY CODES AND THE ACCESS BITS STORED ON THE MIFARE CARD ARE NOT CHANGED UNTIL THEIR OPERATION IS FULLY UNDERSTOOD.

Sector Trailer Block



KeyA and KeyB

The security Key codes are each six bytes long and for successful authentication and read/write communication the NFC RFID Reader device would have to have one or both keys stored in its internal memory as well (depending on Access Bit settings). To allow "out-of-the-box" operation with new Mifare transponders and NFC RFID Reader devices, Transport Key values are pre-loaded into the transponder memory and also into the NFC RFID Reader memory.

Transport Key values as defined by Infineon are:

KeyA: 0xFF FF FF FF FF FF **KeyB:** 0xFF FF FF FF FF FF

Transport Key values as defined by Philips Semiconductors are:

KeyA: 0xA0 A1 A2 A3 A4 A5 **KeyB**: 0xB0 B1 B2 B3 B4 B5

These KeyA and KeyB values are stored as repeated pairs in the NFC RFID Reader MF device as a factory default.

If the user intends using other Mifare cards then they may have different transport keys loaded so the user would have to use the STORE KEY command to load the particular Key codes into the NFC RFID Reader memory first. Once correct communication is established then Keys and Access bits can be changed on the card and NFC RFID Reader to suit the users final system requirements.

Access Bits

The access conditions for every data block and sector trailer are defined by three bits (C1, C2, C3), which are stored in a specific non-inverted and inverted pattern in the sector trailer of the particular sector. The access bits control the memory access rights using the secret KeyA and KeyB codes. The access bits can be changed provided the relevant Key(s) is known and the current access conditions allow this operation.

Note that with each memory access the internal logic of the Mifare transponder verifies the format of the access conditions. If it detects a format error then the entire sector is irreversibly locked. Since the access bits themselves can be blocked for read/write operations, care must be taken when cards are being personalised for a particular application or service provider.

Access Bits	**Block Affected within a sector**	
	(1k card and lower half of 4k card)	(Upper half of 4k card)

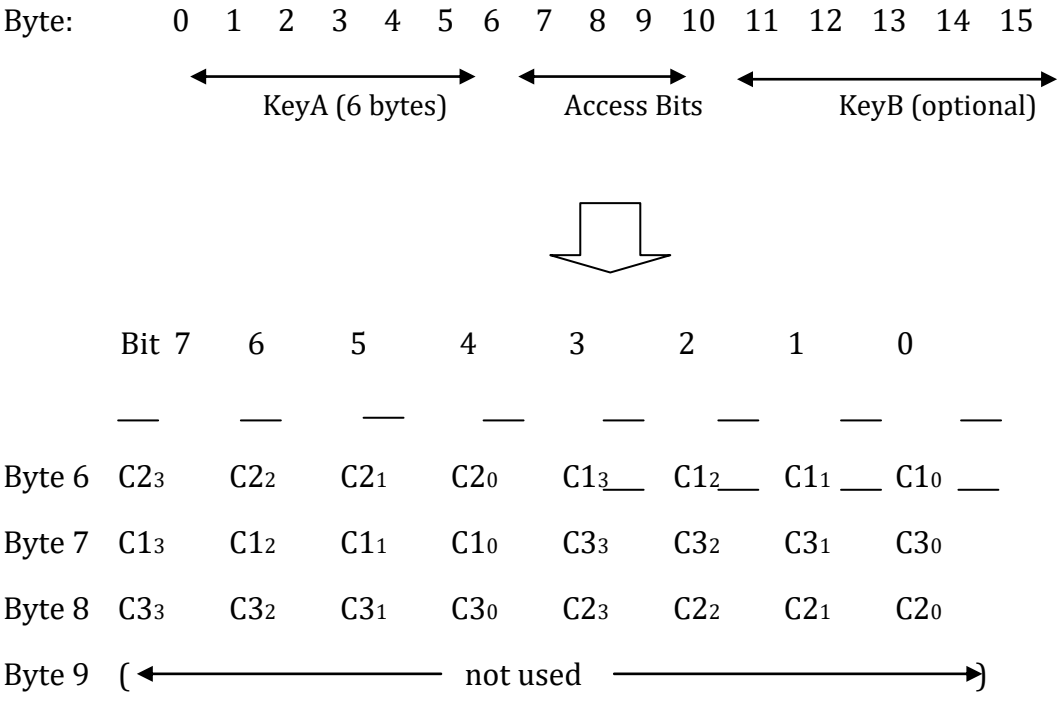
$C1_0$ $C2_0$ $C3_0$	Block 0	(Data Block)	Blocks 0-4	(Data Blocks)
$C1_1$ $C2_1$ $C3_1$	Block 1	(Data Block)	Blocks 5-9	(Data Blocks)
$C1_2$ $C2_2$ $C3_2$	Block 2	(Data Block)	Blocks 10-14	(Data Blocks)
$C1_3$ $C2_3$ $C3_3$	Block 3	(Sector Trailer)	Block 15	(Sector Trailer)

Note that the Access Conditions for individual blocks can be set on 1k cards and for lower half of 4k card memory. However for the upper half of 4k cards the Access conditions are set for groups of five blocks (except for Sector Trailer Block which is still individually set).

Access Conditions

The C1, C2, C3 access bits are stored in a specific non-inverted and inverted pattern in the sector trailer of the particular sector. These bits control the Access Conditions for every data block and sector trailer block.

Sector Trailer Block



Cxy = Inverted bit

Transport settings (factory defaults)

Byte 6	(1	1	1	1	1	1	1	1) = 0xFF
Byte 7	0	0	0	0	0	1	1	1) = 0x07
Byte 8	1	0	0	0	0	0	0	0) = 0x80
Byte 9	(←	not used					→	)

Access Condition for Sector Trailer Block

The read/write access to the Keys and the Access Bits themselves is controlled by the access conditions for the Sector Trailer Block. The read/write access is specified as "Never", "KeyA", KeyB" or Key A|B (KeyA OR KeyB).

Access Bits			**Access condition for:**						
			KEY A		ACCESS BITS		KEY B		
C1	C2	C3	Read	Write	Read	Write	Read	Write	
0	0	0	never	keyA	keyA	never	keyA	keyA	(KeyB can be read)
0	0	1	never	keyA	keyA	keyA	keyA	keyA	**(Transport setting)**
0	1	0	never	never	keyA	never	keyA	never	(KeyB can be read)
0	1	1	never	keyB	keyA|B	keyB	never	keyB	
1	0	0	never	keyB	keyA|B	never	never	keyB	
1	0	1	never	never	keyA|B	keyB	never	never	
1	1	0	never	never	keyA|B	never	never	never	
1	1	1	never	never	keyA|B	never	never	never	

The new Mifare cards have the access conditions predefined as transport configuration:

C1 C2 C3 = (0 0 1) which means Sector Trailer Block can only be read or written to using KeyA and KeyA itself can never be read.

Because the Access Bits themselves can be locked great care must be taken when any of these settings are changed because they may be irreversible making the card unusable.

Access Condition for data areas

The read/write access to the data areas is also controlled by the access conditions defined in the Sector Trailer Block. The read/write access is specified as “Never”, “KeyA”, KeyB” or Key A|B (KeyA OR KeyB).

A data block can be a “read/write block” or a “value block”. For a “read/write” block the basic read and write operations are allowed. For the “value block” the additional increment, decrement, transfer and restore operations can apply. In one case (001) only read and decrement are possible for a “non-rechargeable” card application and in another case (110) recharging is only possible using keyB.

The default transport configuration specifies that the data areas can only be accessed using KeyA|B, however the operation of the Mifare cards define that “IF KEYB CAN BE READ IN THE CORRESPONDING SECTOR TRAILER THEN IT CANNOT SERVE FOR AUTHENTICATION”. This means that for the transport configuration (and 001 and 010 cases), KeyA must be used for access.

Note also that the read-only status of the Manufacturer Block is not affected by the access bits setting.

Access Bits			**Access condition for:**				**Application**
C1	C2	C3	Read	Write	Increment	Decrement, Transfer, Restore	
0	0	0	keyA|B	keyA|B	keyA|B	keyA|B	(Transport setting)
0	0	1	keyA|B	never	never	keyA|B	(value block)
0	1	0	keyA|B	never	never	never	(read/write block)
0	1	1	keyB	keyB	never	never	(read/write block)
1	0	0	keyA|B	keyB	never	never	(read/write block)
1	0	1	keyB	never	never	never	(read/write block)
1	1	0	keyA|B	keyB	keyB	keyA|B	(value block)
1	1	1	never	never	never	never	(read/write block)

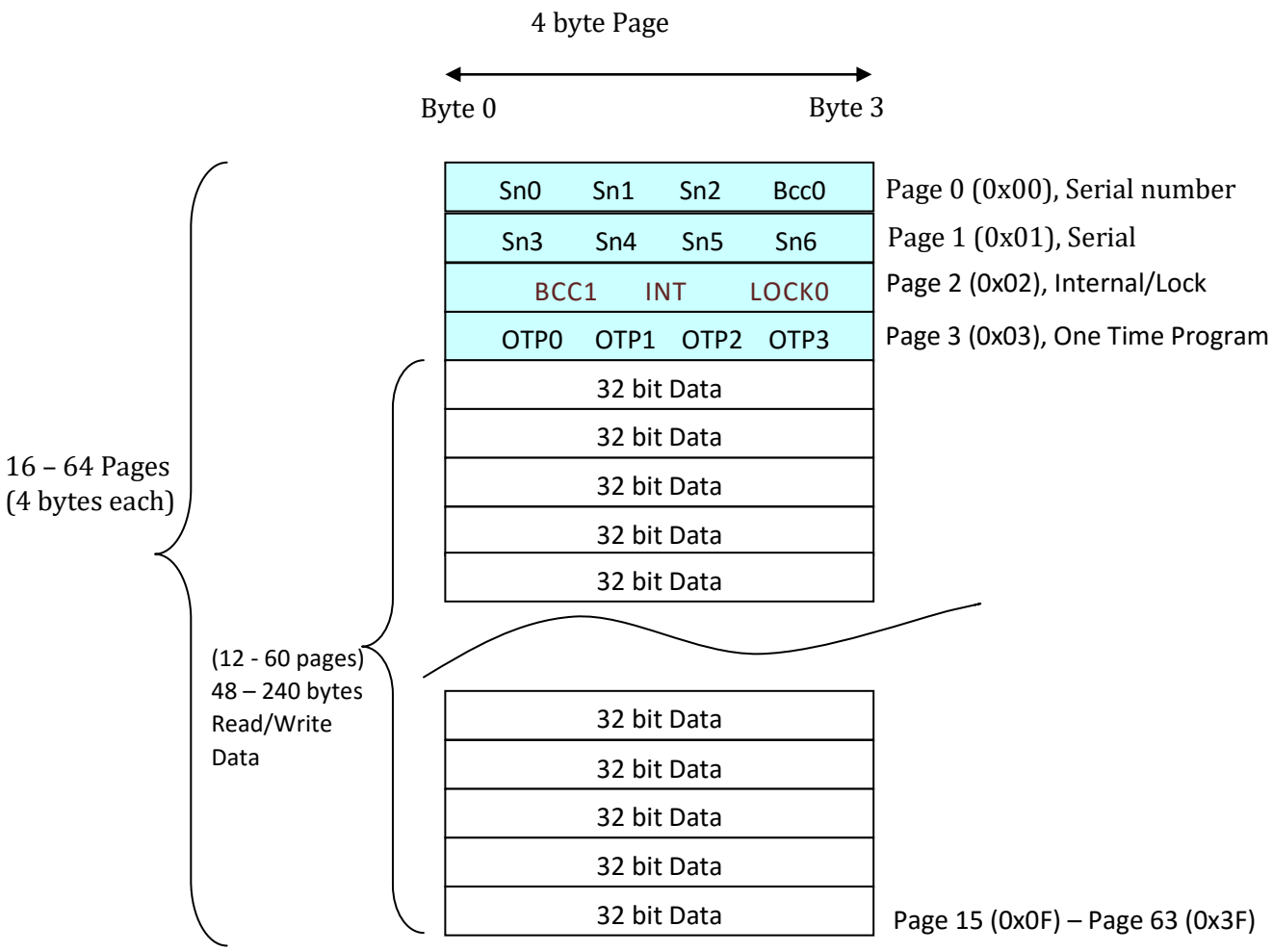
Combining the transport (default) access conditions for the data areas and the sector trailer block, the typical setting is:

Sector Trailer Block

Byte:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	0xA0	A1	A2	A3	A4	A5	FF	07	80	69	B0	B1	B2	B3	B4	B5
	← KeyA (6 bytes) →						← Access Bits →				← KeyB (6 bytes) →					

The default access bits (0xFF 07 80 69) define that KeyA must used for all operations and KeyA can never be read, so in the previous example bytes 0 – 5 (keyA) would always read as zeros.

ULTRALIGHT / NTAG2 (64 – 256 BYTE) MEMORY MAP



Note: Ultralight / NTAG2 card has 7 byte serial number (+ 2 check bytes)

Bcc0 = CT xor Sn0 xor Sn1 xor SN2 (CT = Cascade Tag = 0x88)

Bcc1 = Sn3 xor Sn4 xor Sn5 xor Sn6

The Ultralight / NTAG2 transponder has a different structure to the Mifare 1k and 4k transponders. The 64 - 256 byte memory is organised as 16 - 64 pages of four bytes each. The first two pages (and byte 0 of page 3) contain a seven byte serial number (UID) together with two check bytes. The other "system" bytes are used for locking card features and providing a number of OTP (One Time Programmable) data bytes. The remaining 12 – 60 pages (48 - 240 bytes) can be used for general read/write storage.

Despite having a different memory structure, the Ultralight / NTAG2 uses the same Mifare communication protocol except that a cascaded selection procedure is used and the **Authentication procedure is not used for read/write operations**. The security Keys are therefore NOT required for any operations to an Ultralight /NTAG2 card. Because of its smaller memory, more basic operation and lower cost, the Ultralight /NTAG2 card is typically used for simpler applications.

The NFC RFID Reader NFC reader module fully supports the Ultralight / NTAG2 cards and the read and write commands have the same format as for the Mifare cards except that dummy data bytes (0x00 values) are used for the Key number etc. In addition the "block number" argument becomes a "page number" parameter.

MIFARE APPLICATIONS AND SECURITY

The Mifare "classic" family is the pioneer and market leader in contactless smart card technology operating in the 13.56 MHz frequency range with read/write capability. The Mifare technology was originally designed for Electronic-Purse applications for public transport systems and was a benchmark for the ISO 14443A standard. The cards have up to 40 separate memory sectors or "purses" (on Mifare 4k card) that can be individually locked and unlocked for access. In addition, the high speed communication of the 13.56 MHz interface allows for quick increment/decrement operations that are required for rapid ticketing or e-purse applications. Typical transaction time is less than 100ms for read/modify/write operations. The multiple memory sectors allow for different service providers to use the same Mifare card with complete independence and security, the Ultralight /NTAG2 cards do not have separate memory sectors and are designed for simple applications requiring low cost.

Because Mifare was designed originally for Electronic-Purse applications, a high level of security was essential to prevent fraud. Each transaction is started with a mutual three pass authentication

procedure according to the ISO 9798-2 standard. RF communication is protected from replay attack and data communication between NFC RFID Reader and card is encrypted according to the Philips/NXP CRYPTO1 algorithm. Separate sets of two security keys for each memory sector (or application) ensure that service providers have complete security control over their individual sector. The high level of data integrity for the 106 kbaud RF communication is achieved by combining a special patented modulation technique, 16 bit CRC, parity bit coding, bit counting, channel monitoring and an anti-collision algorithm.

Finally, the NFC RFID Reader device itself stores up to 32 security keys that the service provider can change and use for their applications. These keys are non-volatile and cannot be read back further ensuring system security.

For additional security the NFC RFID Reader has a built-in and configurable “Dynamic” software encryption system, this uses the cards unique serial number (UID) as a seed to the algorithm so the same data stored on multiple cards would always appear as different scrambled data.

The Mifare, Ultralight / NTAG2 cards each have a unique serial number and with both hardware and software encryption features can be used for almost any application requiring large and secure read/write memory areas. Typical applications could include:-

Access Control, Automatic Fare Collection, Bus/Train/Airline ticketing, Electronic Purse,

Vending, Loyalty / Membership Schemes, ID cards, Time and Attendance, Asset Tracking, Gambling, Electronic Keys, Logistics, Road Tolling, Payphones, Park and Ride Schemes, pre-paid metering.

IB Technology’s design philosophy has created the “Universal RFID Socket” so that different technologies such as the 125 kHz NFC RFID Reader (Hitag and EM Marin) modules, QT (Quad Tag) module and the 13.56 MHz NFC RFID Reader Mifare and I.CODE modules are the same physical size, have the same pinout and share common serial commands. This concept allows users to select and migrate between different RFID technologies according to the specific features they require with the minimum of design changes.

(Hitag, Mifare and I.CODE are registered trademarks of NXP/Philips Semiconductors N.V)

NFC RFID READER (LOW-POWER) SPECIFICATION

The NFC RFID Reader NFC LOW-POWER version is a complete read/write system for 13.56 MHz NFC Type 2 (Mifare 1k, 4k, Ultralight / NTAG2) cards and tags and serial number acquisition from NFC type 4 cards. The module is pin-compatible and similar in operation and features to the “standard” MIFARE version (NOTE differences in EEPROM parameters and Auxiliary output formats).

The LOW-POWER version uses a different specification microcontroller offering lower voltage operation and can be powered from four alkaline battery cells. During the Polling Delay period the

microcontroller enters SLEEP mode with the RF device in hard power-down mode to reduce the current consumption to a very low level.

The module wakes up after the polling delay period and the process repeats. The NFC RFID Reader-MIFARE "low-power" Windows applications can be used to configure the parameters and read/write data.

PARAMETER	TYPICAL VALUE
Supply Voltage (performance optimised for 5 volt operation)	4 – 6 volts DC (operation from 4 x alkaline cells)
Operating temperature	-40 deg C to + 85 deg C
Minimum AVERAGE current consumption. (1+ second polling)	**approx 100 µA**
Active period for RF AND Pi communication (each polling cycle).	Up to 20 mS
Peak antenna voltage (optimum tuning)	30 volts peak-to-peak
Peak antenna current (optimum tuning) for short period each polling cycle (up to 10 mS burst)	150 mA
Polling Delay (SLEEP / Power-down mode)	4 mS to 8 seconds
Current consumption during Polling delay / SLEEP	Less than 20 µA
Current consumption during RF ON each polling cycle	Less than 20mA
Maximum data rate (between card and NFC RFID Reader)	106k baud
Range (dependent on tag size, antenna dimensions and tuning)	25-80 mm
Auxiliary output drives	Up to 25mA
Serial Interface	TTL level RS232
Serial Communication Parameters	9600 baud, 8 data bits, no parity, 1 stop bit protocol with Command Strobe handshake

Basic electrical specification with LEDs pins and auxiliary outputs NOT connected.

Note that the NFC RFID Reader is designed for optimum performance and range at 5-volt operation. Performance will be reduced at maximum and minimum operating voltage.

RASPBERRY PI DRIVER SOFTWARE

Example software for the Raspberry Pi is available on www.Bostintechnology.com.