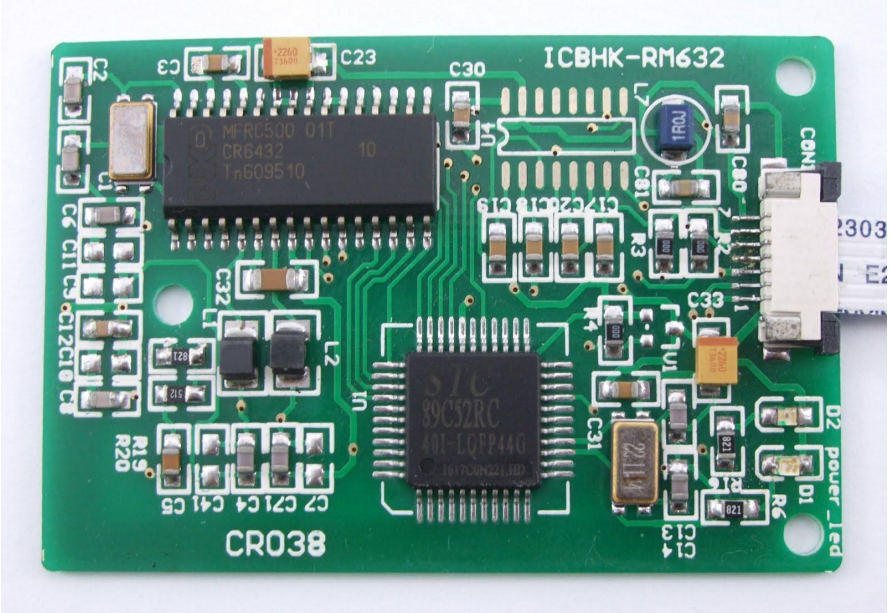
MIFARE Reader CR038
RFID-ICRW-CR038



User's Manual

V1.0

Jan 2013

Index

1. INTRODUCTION

NFC (Near Field Communication). It is a short-range radio technology that enables communication between devices that either touch or are momentarily held close together. It is based on and extends on RFID and it operates on 13.56 MHz frequency. Mifare was one of it. BTW, NFC is an open-platform technology, and is being standardized by NFC forum. For more information, check the Introduction to NFC by Nokia. This document will explain the usage of Mifare card/tag reader/writer. The CR038A Mifare Reader/Writer is a device to read and write information from/to Mifare card, Classic 1K or 4K.

Features:

- +5VDC operate
- Current consumption: 80 to 100mA
- Interface: TTL UART at 19200bps, 8-N-1
- R/W (Read/Write) distance: Upto 40mm, depends on tag
- Storage temperature: -40ºC ~ +85ºC
- Operating temperature: 0 ºC ~ +70ºC

Currently, the Mifare card we are carrying is ISO14443A 1K Classic card:

- 1024 bytes EEPROM, divided into 16 sectors with 64 bytes on each sector
- 100,000 write endurance cycles
- 10 years data retention
- ISO 14443 A
- 13.56MHz transponder frequency
- 106 kbit baud rate
- Bit-wise anti-collision
- Up to 10 cm operating distance (depends on reader/writer)
- 4 bytes non-unique serial number (NUID)
- 2 bytes access key per sector
- Individual access condition for each sector

2. PACKING LIST

Please check the parts and components according to the packing list. If there are any parts missing, please contact us at sales@cytron.com.my immediately

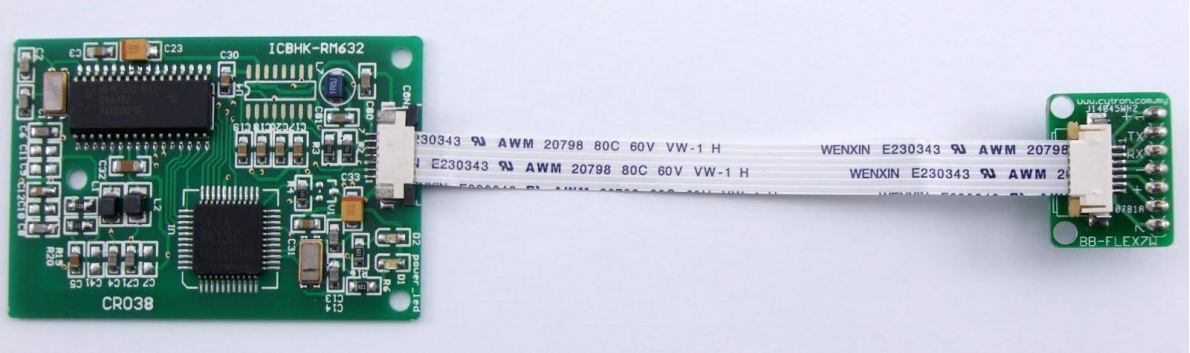
Packing list:

1. Mifare RFID Reader/Writer x 1
2. Breakout Board x 1
3. 7-ways straight header pin (not soldered)

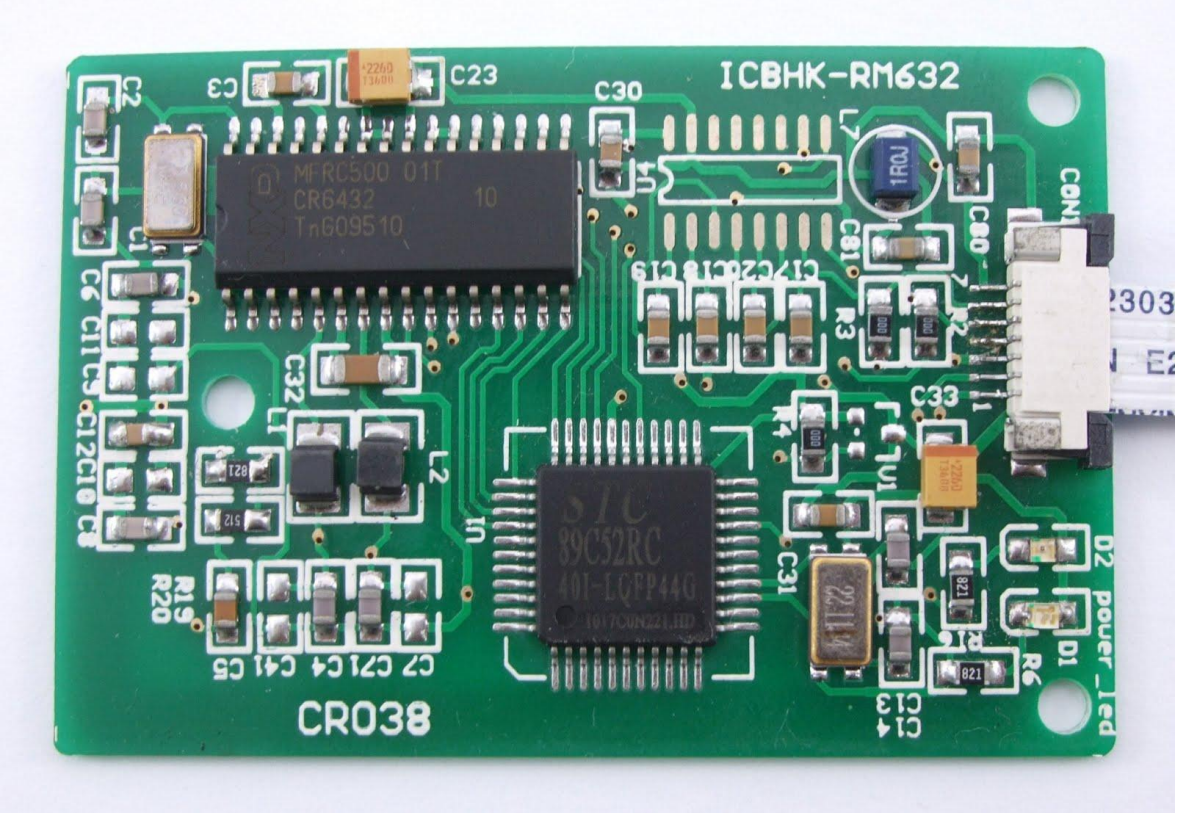
Optional/Recommended Items: (Please get it separately from Cytron online store)

- UC00A or UC00B - To ease the connection/interface to computer or laptop.

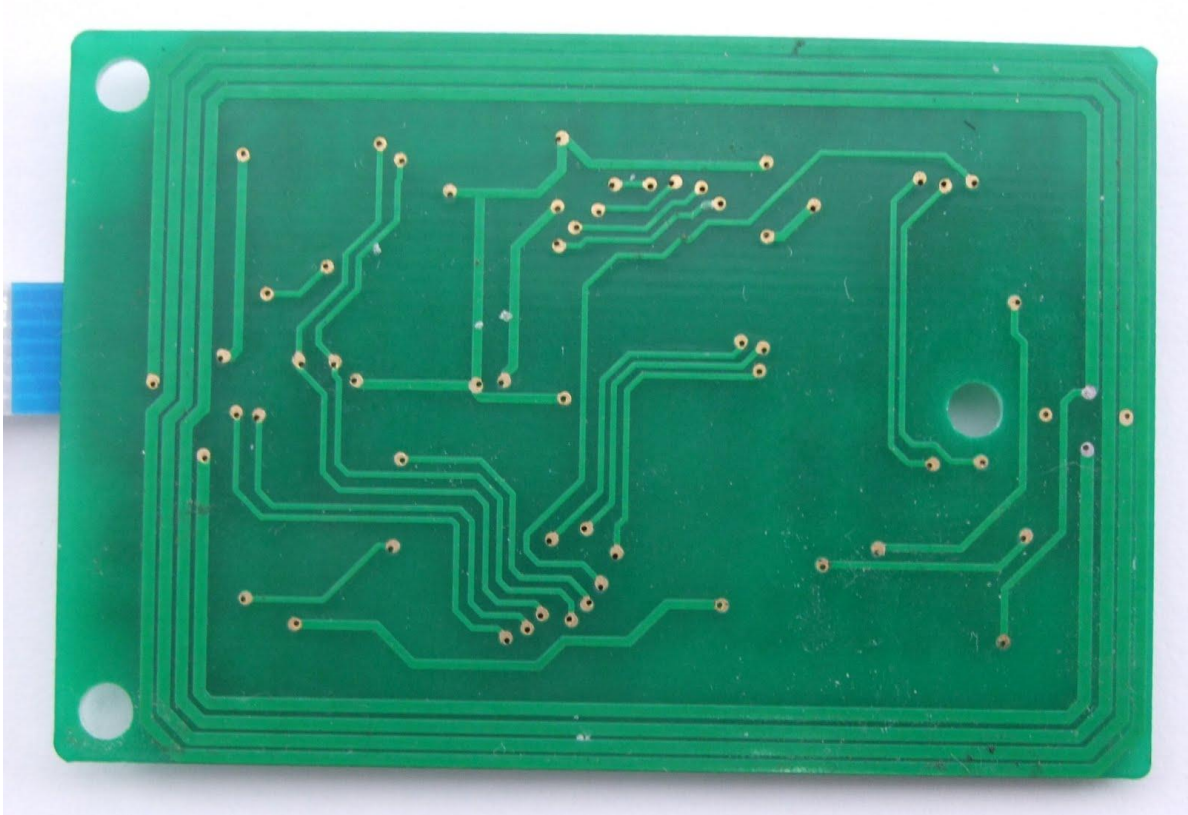
3. BOARD LAYOUT



CR038A board with flex cable and breakout board (header pin is not soldered)



CR038A board (top view)



CR038A board (bottom view), PCB antenna

Flex cable breakout board (header pin is not soldered)

4. How Does It Works?

How does RFID work? A **R**adio-**F**requency **ID**entification system has three parts:

- A scanning antenna
- A transceiver with a decoder to interpret the data
- A transponder - the RFID tag - that has been programmed with information.

There are many standards of RFID in the market. This particular reader/writer is compatible with Mifare Classic cards which normally come in 1K or 4K bytes of EEPROM built-in. There are many terms being used for RFID, you must have the correct reader/writer and transponder/tag to get the communication up. The purpose is for reader to read data from transponder/tag and also for writer to write information into the transponder/tag. The CR038A is a reader and writer and it support Mifare Classic 1K or 4K card with uses the standard protocol of ISO14443A, operating at 13.56MHz frequency. **Do not** get confused with another RFID device we are carrying, The passive RFID tag and reader. Mifare refers to card/tag/transponder defined by NXP (formerly known as Philips) is a standard subset from NFC (Near Field Communication), using 13.56MHz, readable/writable. It follow the standard protocol of ISO 14443, the protocol from radio layer up to command layer. For more information, please visit Mifare (Wiki).

Mifare Card/Transponder/Tag

Before we talk about the Reader/Writer, let's discuss about the card. Mifare require:

- a device that able to read and write;
- antenna to transmit power, communicate and also receive power;
- and lastly the transponder or tag, sometime is called card.

The Mifare Card that Cytron carry is Mifare Classic 1K, ISO14443A. It has 1K (1024) bytes of EEPROM memory in it. User can access to these memory using this RFID reader/writer.

Mifare Classic 1K card divided the memories into segments called sectors. 1K card has 16 sectors, with each sector have 4 blocks, and each block have 16 bytes of memory. Each sector of memory is protected by two different keys, called A and B. Each key can be programmed to allow access operations such as reading, writing, increasing value blocks, etc. However, 16 bytes of each sector (1 block) are reserved for key A, B and access conditions; and cannot normally be used for user data. Not to forget, the 1st 16 bytes (block 0) contain the serial number of the card (4 bytes) and information about the card manufacturer data and read only. That will end up the actual usable memory for 1K Mifare Classic card to 752 bytes. That is quite some space to store value, names, biodata, time, etc.

Memory Organization (1K)

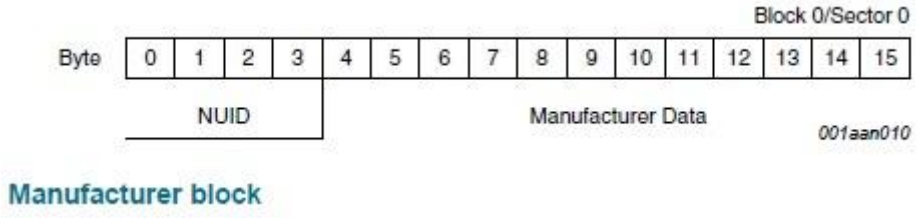
Here is the table showing the memory organization of Mifare Classic 1K Tag.

Sector	Block	Byte Number within a Block 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Description
15	3	Key A						Access Bits				Key B						Sector Trailer 15
	2																	Data
	1																	Data
	0																	Data
14	3	Key A						Access Bits				Key B						Sector Trailer 14
	2																	Data
	1																	Data
	0																	Data
:	:																	
:	:																	
:	:																	
1	3	Key A						Access Bits				Key B						Sector Trailer 1
	2																	Data
	1																	Data
	0																	Data
0	3	Key A						Access Bits				Key B						Sector Trailer 0
	2																	Data
	1																	Data
	0	Manufacturer Data																Manufacturer Block

Memory organization of Mifare Classic 1K Transponder/Tag/Card

There are 16 sectors, starting from sector 0, and each sector will have 4 blocks, which start from block 0 to block 3. Each block will have 16 bytes of memory. The access to data on each sector is controlled by the value of last block, which is block 3 (sector trailer) of each sector. Normally the block is being number from 0 to 63, user will need to decide which sector he/she wanted to access and provide the correct Keys and access bit. Block 3 (Sector Trailer) of each sector store 2 keys value and access bits. It is named as Key A and Key B, both are 6 bytes and the Access bits are 4 bytes (actual is 3 bytes).

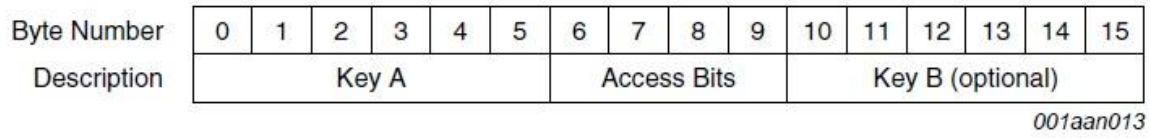
Block 0 of sector 0 (1st data block of 1st sector) is manufacturer block as is it readable only because it is being programmed by manufacturer during production and is One Time Programmable only.



Manufacturer block (Block 0 of Sector 0) in Mifare Classic 1K

Therefore, there are 3 user data blocks in each sector, except 2 user data block for sector 0 because of manufacturer block. Now, you need to know how to access these data blocks? Sector trailers hold the keys.

Sector trailer is block 3 of each sector, there are 3 portion of data in it. We know each block is 16 bytes, sector trailer too. 6 bytes is allocated for Key A (front), another 6 bytes for Key B (back), middle 4 bytes are for Access bits, total 16 bytes!



- Keys A and B (optional), which return logical "0"s when it is read
- Access bits control and decide the access for blocks of that sector (including sector trailer). It is stored in byte 6 to 9. The access bits also specify the type (data or value) of the data blocks.

In some cases, if Key B is not needed, the last 6 bytes of the sector trailer can be used as data bytes. The access bits for the sector trailer have to be configured accordingly. Byte 9 of the sector trailer is available for user data (1 byte). For this byte the same access rights as for byte 6, 7 and 8 apply. But normally we do not access (change) these bytes because changing the value of access bytes (byte 6, 7 and 8) incorrectly will permanently lock that particular sector (4 blocks of memory).

By default, when sector trailer is read, the key (A & B) bytes are blanked out by returning "0"s. If Key B is configured to be readable, the data stored in bytes 10 to 15 will be shown when it is read. Also, by default or sometime is called transport configuration, Key A must be used for Authentication, or the password for you to access (read or write) the data. Sector trailer can be read and written as other block, but bear in mind, writing incorrect value to Access bits and Keys might permanently lock the sector, losing all 4 blocks of memory. Please do refer to MF1S50x datasheet for more details on Key A, Key B and Access bit. We will not cover these in detailed here. We will focus on the steps to read and write data into

Mifare Classic 1K card. Just take an example, you want to read data from block 10 (block 2 of sector 3).

1. Initialize the CR038 device.
2. Activate the antenna for further operation. The tag require power from device antenna.
3. Request Mifare Standard. CR038 will reply with card type that readable from the antenna. The Mifare card must be on the CR038.
4. Anti-collision loop. The NUID for Mifare card will be read by CR038 and return to host.
5. Select card with the NUID you just get from previous step.
6. Now you can authentication the particular sector with Key A or Key B. Default is using Key A, and the default key is FF FF FF FF FF FF (hexadecimal value).
7. If authentication is successful, you can now read or write data from/to blocks within the same sector of that Mifare card. In this case is block 10. Block 8,9 and 11 can also be access because they are in same sector. Do take note that block 11 is the sector trailer.

The detail protocol of CR038 will be discussion in later section.

Sector Trailer

Well, we will need to highlight that sector trailer is the most important block in Mifare card. It holds the roles of security (password), and how reader/writer access to the memory, how the memory is being translated. So please think twice before making any write operation to sector trailer.

Death Sector: Currently, the CR038 uses the low level read and write to access the MIFARE card. There is no filter or whatsoever mechanism to check whether the Access bits you write are correct, or whether you are writing to correct block. Therefore, a single wrong bit might result in death sector, further lose all the data in it, permanently. Same apply if you lose the Authentication key, because without the correct Key, you are unable to read or write, losing the whole sector too.

Default sector trailer (Transport Configuration)

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Key A						Access Bits				Key B					
FF	FF	FF	FF	FF	FF	FF	07	80	69	FF	FF	FF	FF	FF	FF

Key A: is not readable (return 0 when be read), can be changed and has all power (read, write, increment, decrement).
Key B: is readable therefore can't serve as authentication.

Changing the Access bits, change the authentication.

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Key A						Access Bits				Key B					
						08	77	8F	00						

Key A: is not readable, can be changed and has power to read and write block
Key B: is not readable, can be changed and has all power

Access Bits:
All MiFare Cards have a fine grained access rights system. Each sector can be secured using two different keys (Key A and Key B). Using access bits, it allows read or write access to one or both of the keys for each block. That means, that e.g. you can use Key A in your customer application which is only able to read the data, but use Key B in your internal application to initialize the cards with full write access.

To identify the access rights for a sector there are three bits, called access bits C1, C2 and C3. With these three bits eight different modes are possible. C1 is the LSB.

There are four access rights per sector (one for each three data blocks and one trailers block), so each block at MiFare 1K and the lower 32 blocks at MiFare 4K has its own three access bits. Depending on whether you set the access bits of a data block or of a trailer block (the fourth block of each sector) these bits change their meaning.

When writing the access bits of a data block you can define the following things for this block(this setting is called "block mode".

- Is the data block readable/writeable and by which key (Key A or Key B or both)
- Is it a value block or a read/write block
- Is the block locked (not read/writeable)

The Access Condition for Data Blocks

Access bits			Access condition for				Application
C1	C2	C3	read	write	increment	decrement, transfer, restore	
0	0	0	key A|B[1]	key A|B1	key A|B1	key A|B1	transport configuration
0	1	0	key A|B[1]	never	never	never	read/write block
1	0	0	key A|B[1]	key B[1]	never	never	read/write block
1	1	0	key A|B[1]	key B[1]	key B[1]	key A|B[1]	value block
0	0	1	key A|B[1]	never	never	key A|B[1]	value block
0	1	1	key B[1]	key B[1]	never	never	read/write block
1	0	1	key B[1]	never	never	never	read/write block
1	1	1	never	never	never	never	read/write block

A|B means Key A or Key B can be the authentication key
NEV = Never, is not allowed.
[1] If Key B may be read in the corresponding Sector Trailer it cannot serve for authentication (all grey marked lines in the table). As a consequences, if the reader authenticates any block of a sector which uses the grey marked access conditions and using Key B, the card will refuse any subsequent memory access after authentication.

Example, although a new card allow authentication using Key B, which is in mode 0 (transport configuration), it does not allow further any access (read or write) after the authentication process is done. Because in transport configuration in sector trailer, Key B is readable. Therefore, a new card must use Key A for authentication.

How about the Key A and Key B, also the Access bit? Can user change it? Yes, it is allowed, the sector trailer block has its own block mode, it is named as Access Mode. Let's look at the Access modes.

The Access Condition for Sector Trailer

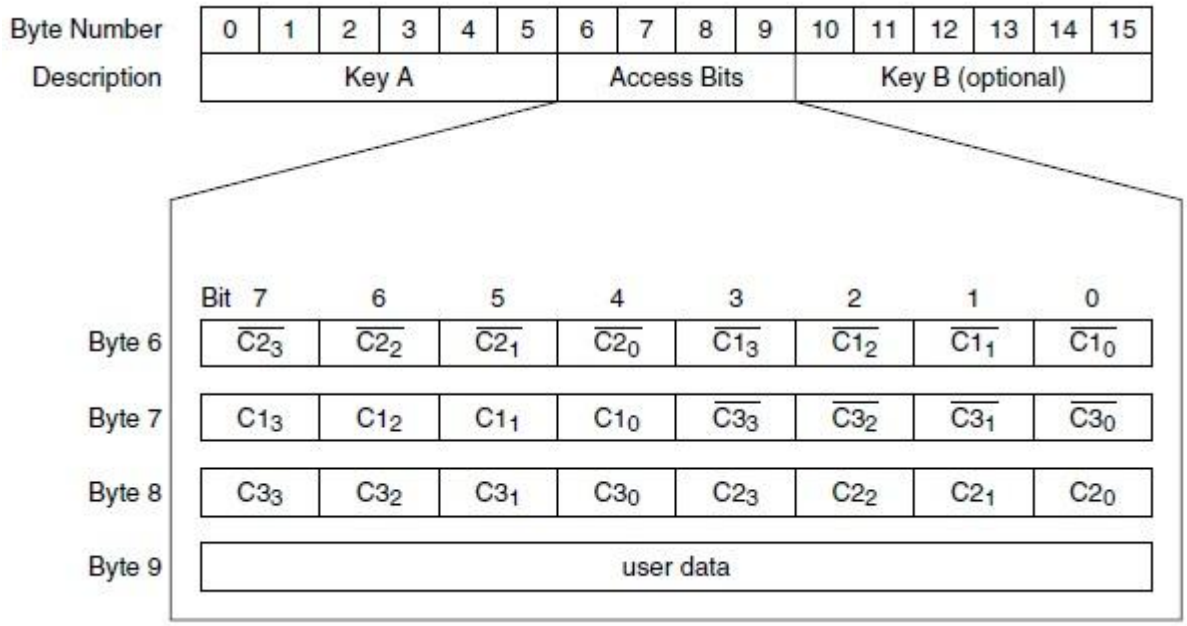
Access bits			Access condition for						Remark
			KEYA		Access bits		KEYB		
C1	C2	C3	read	write	read	write	read	write	
0	0	0	never	key A	key A	never	key A	key A	Key B may be read[1]
0	1	0	never	never	key A	never	key A	never	Key B may be read[1]
1	0	0	never	key B	key A|B	never	never	key B	
1	1	0	never	never	key A|B	never	never	never	
0	0	1	never	key A	key A	key A	key A	key A	Key B may be read, transport configuration[1]
0	1	1	never	key B	key A|B	key B	never	key B	
1	0	1	never	never	key A|B	key B	never	never	
1	1	1	never	never	key A|B	never	never	never	

[1] For this Access Condition, Key B is readable and can be as user data.

Example, Access-Mode 4 which is transport configuration (new card): In this mode the access bits can only be read or written when using A for authentication. The details of Mifare card format, please refer to MF1S503x document from NXP. That is good and detailed explanation, including the access bits, key A and B. procedures to read and write data.

How is these Access bits being stored? Each memory block have 3 bits, there are 4 blocks in a sector, so there should be 12 bits, right? But for data integrity purpose, it is being stored in 3 bytes (18 bits). Access bits are located from byte 6 to byte 8 in sector trailer of each sector. It keeps 4 blocks access bits. However, some access bits are duplicated with inverted condition.

Access bits Organization:



Confused? Just take some time to study it and you will understand. You will need to extract it into bits.

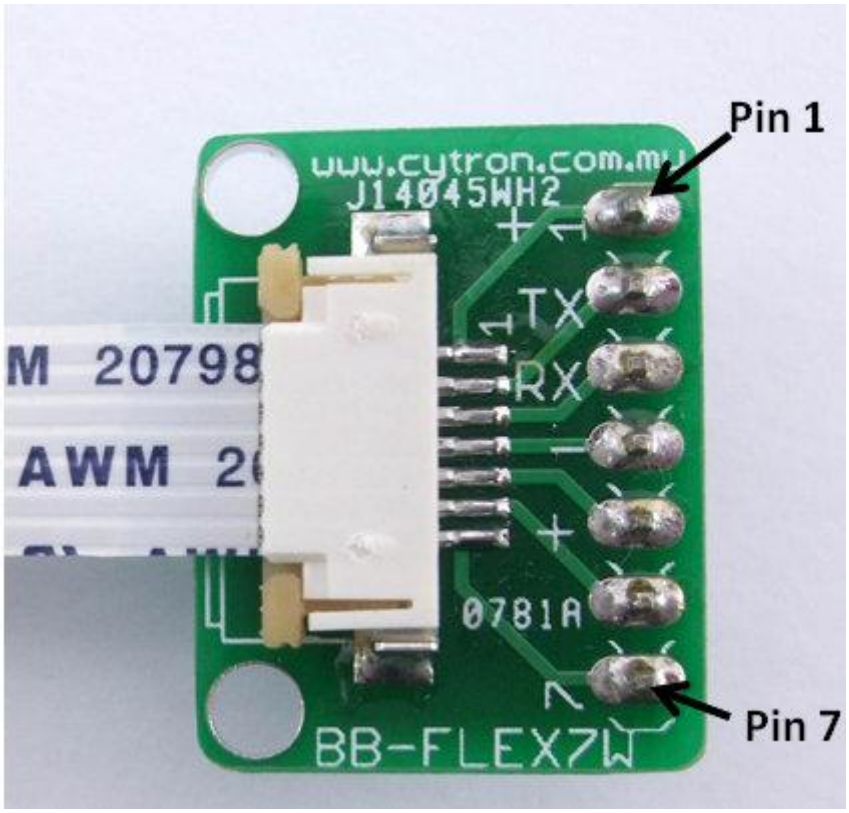
NOTE: Do double check the data you write into sector trailer, a single wrong bit might locked the whole sector (4 block of memory) permanently.

5. HARDWARE INTERFACE

The CR038 Mifare reader/writer uses 5V TTL UART to communicate with host. The host can be a computer or microcontroller. The UART settings are:

- **Baud rate**: 19200 bps (default)
- **Data**: 8 bits
- **Stop**: 1 bit
- **Parity**: None
- **Flow control**: None

The small flex-cable breakout board allow easy access to CR038. There are only 5 meaningful pins on the breakout board.



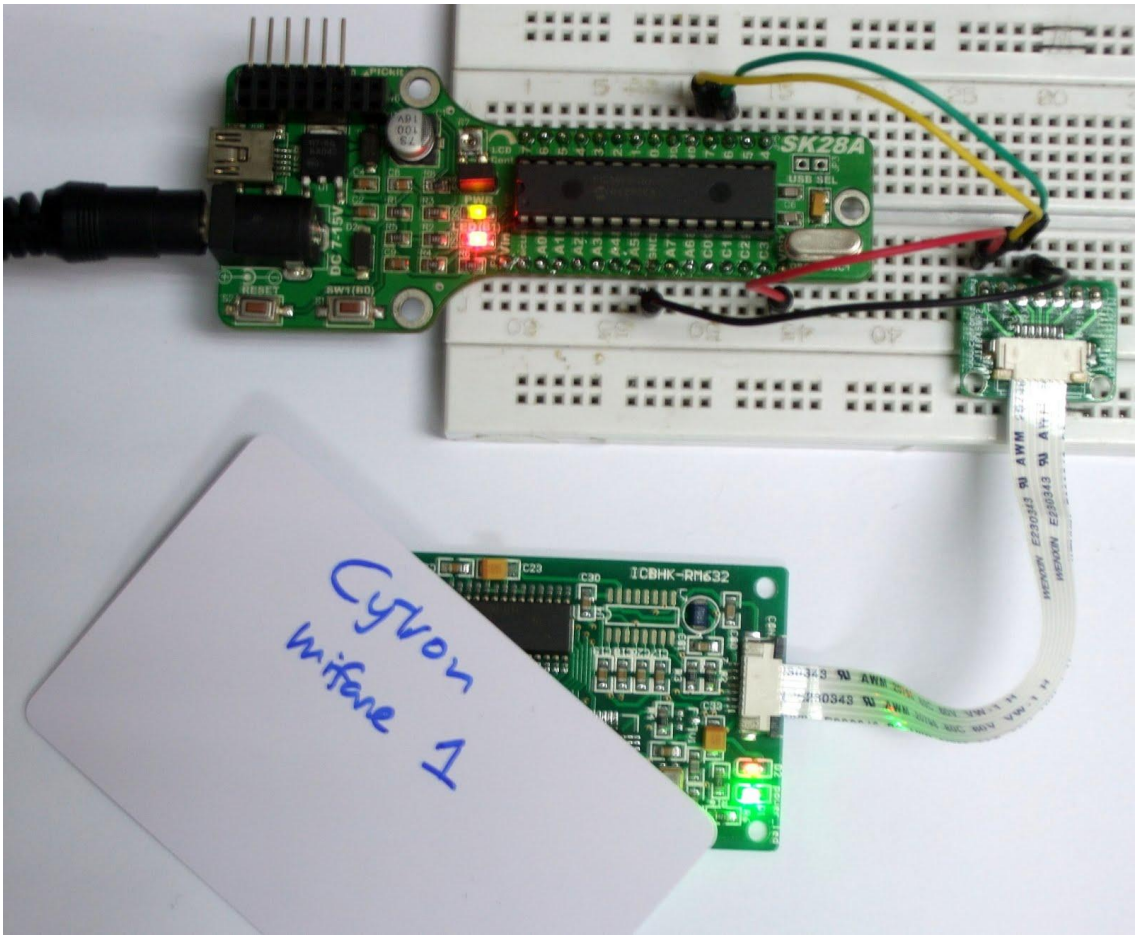
pin-out of CR038

Pin-out of CR038 breakout board

Pin No.	Function	Remarks
1	+ve power, 5VDC	positive supply to CR038 board
2	UART's TX, transmitter of CR038	Connect to RX of host, example microcontroller's RX pin, 5V TTL
3	UART's RX, receiver of CR038	Connect to TX of host, example microcontroller's TX pin, 5V TTL
4	-ve power, 0V, GND	Ground of the power and signal
5	+ve power, 5VDC	positive supply to CR038 board

Therefore, pin 1 and 5 are shorted, and of course connecting either one of these pin to +5VDC is enough to power the board.

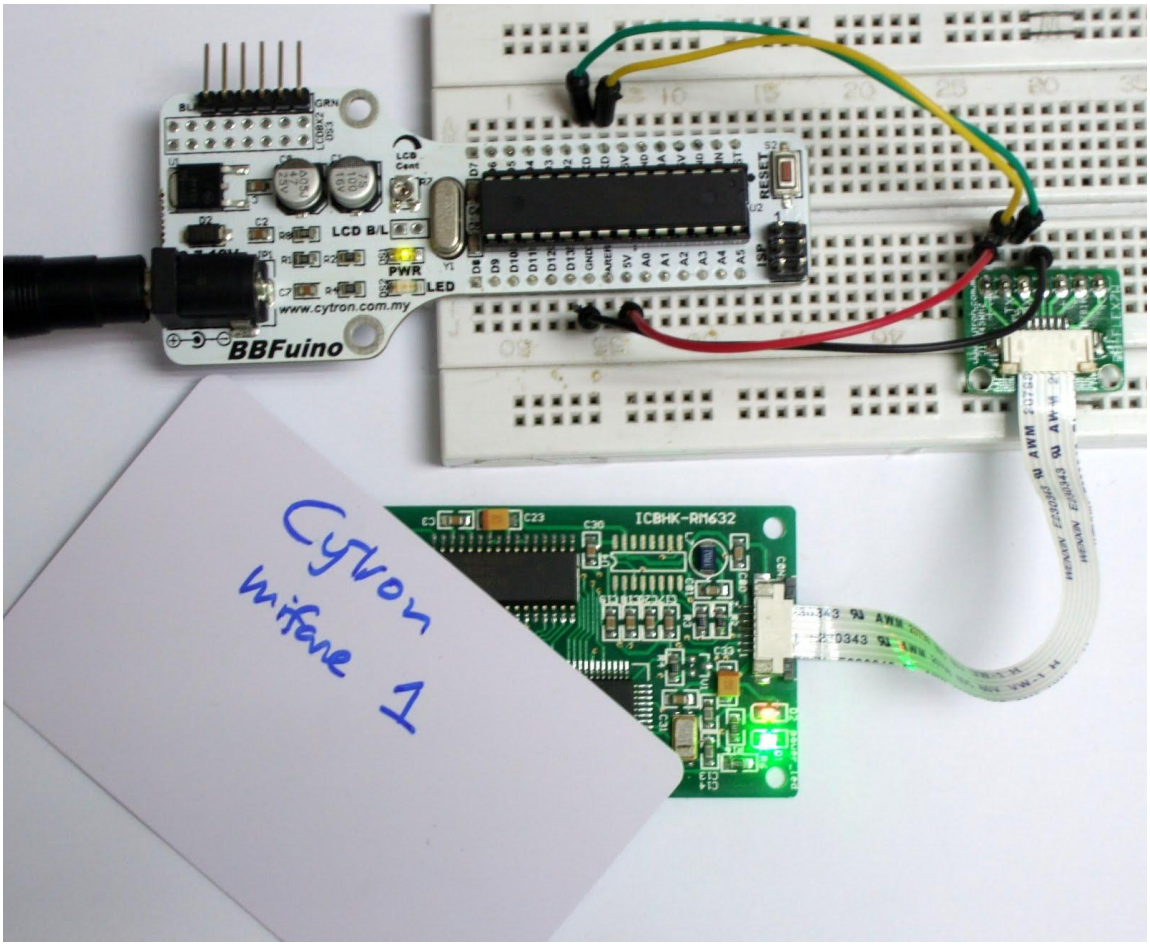
SK28A with PIC16F876A interface with CR038



Connections:

SK28A + PIC16F876A	**CR038 Mifare Reader/Writer**
VDD (5V)	+ (pin 1 or 5)
GND (0V)	- (pin 4)
RC7 (RX)	TX (pin 2)
RC6(TX)	RX (pin 3)

BBFuino with CR038

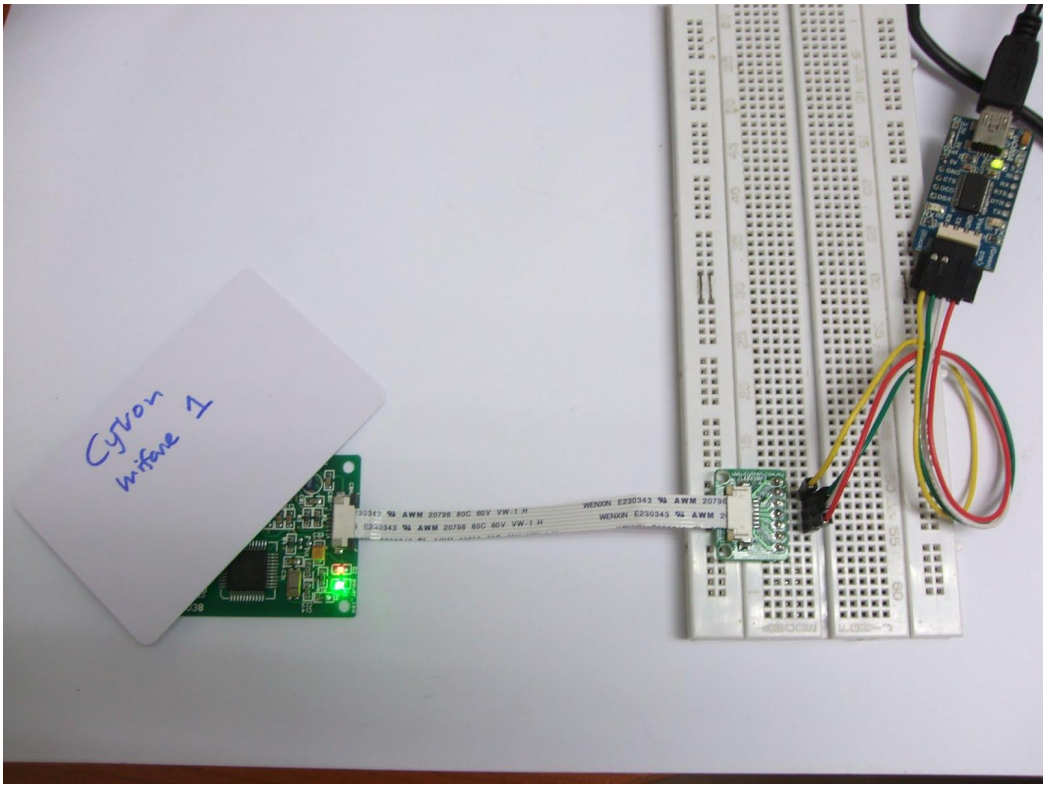


Connections:

BBFuino	**CR038 Mifare Reader/Writer**
5V	+ (pin 1 or 5)
GND (0V)	- (pin 4)
RXD	TX (pin 2)
TXD	RX (pin 3)

Note: BBFuino and most of Arduino main board (Duemilanove, UNO, PRO-Mini) uses UART to load program, CR038 also uses UART to communicate. Therefore, during loading sketch from computer to BBFuino, the UART connection with CR038 need to be disconnected.

Computer with CR038 using UC00A



Connections:

UC00A	**CR038 Mifare Reader/Writer**
Vsel (5V)	+ (pin 1 or 5)
GND (0V)	- (pin 4)
TX	RX (pin 3)
RX	TX (pin 2)

6. SOFTWARE (PROTOCOL)

CR038 uses packetized UART communication. This means command/functions and results are communicated in several bytes of data with header, length, data, also checksum.

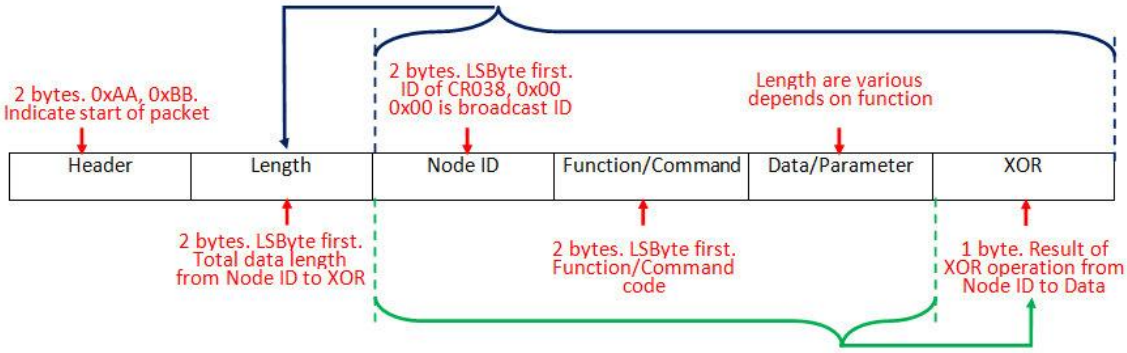
Data Packet

Transmission Rate : Default 19200.N.8.1
Data Format : HEX

Data Packet. The packet general contains header, followed by length, Node ID (CR038 ID number), function/command Code, the data or parameters, lastly the checksum, it uses XOR. The CR038 will only execute a command/function if and only if it receive whole packet of data correctly. It must comes with 2 bytes of header 0xAA, and 0xBB, follow by 2 bytes of data length, Node ID, Function Code, Data (optional) and result of XOR. The XOR is result (1 byte) from performing exclusive OR operation from Node ID byte until last byte of Data.

Packet format:

Header	Length	Node ID	Function Code	Data	XOR



Packet from Host to CR038, the command:

Data	Data Length (Byte)	Remarks
Header	02	Fixed : 0xAA follow by 0xBB.
Length	02	Length of Node ID to DATA, lower byte first.
Node ID	02	Destination Node ID. This is CR038 ID. Low byte first 00 00: Broadcast ID.
Function/Command Code	02	Function or Command code. Lower byte first.
Data/Parameter	00~0xD0	Data or parameter needed for function/ command. It can be 0 or maximum is 208 bytes. length depends on function.
XOR	01	result from XOR operation each byte from from Node ID to Data.

Packet from CR038 to host, the event :

Data	Data Length (Byte)	Remarks
Header	02	Fixed : 0xAA follow by 0xBB.
Length	02	Length of Node ID to DATA, lower byte first.
Node ID	02	Node ID of CR038, lower byte first.
Function/Command Code	02	Function/Command code received from host. To indicate this packet is reply on the function sent.
Status	1	1 byte, function/command result, 0 = Success, Not 0 = fail.
Data/Parameter	00~D0	Data or parameter return from CR038 based on function/command. It can be 0 or maximum is 208 bytes. length depends on function.
XOR	01	XOR byte from Node ID to Data

Note 1: If any byte from "Length" to "XOR " has value of "0xAA", there should be an extra byte of "0x00" follow it, but value of "Length" does not changed. This is to differentiate it with 0xAA in the "Header".

Note 2: If a command sent (from host to CR038) and there is no reply from CR038 after 100ms, it is consider this command failed. Normally if "Length" or "XOR" is not correct, there is no reply from CR038.

Function/Command Set	Code
• Initialize Port	0x0101
• Set LED Status	0x0107
• Set Antenna Status	0x010c
• Mifare Card Type Request	0x0201
• Mifare Anti-collision	0x0202
• Mifare Select Card	0x0203
• Mifare Halt	0x0204
• Mifare Authentication2	0x0207
• Mifare Read	0x0208
• Mifare Write	0x0209
• Mifare Initval	0x020A
• Mifare Read Balance	0x020B
• Mifare Decrement	0x020C
• Mifare Increment	0x020D
• Mifare Restore	0x020E
• Mifare Transfer	0x020F

Initialize Port: 0x0101
Function: Set Baud Rate of CR038's UART

Packet Format (Hexadecimal): AA BB 06 00 00 00 01 01 Baud XOR
AA BB = Header, 2 bytes.
06 00 = Packet length, 2 bytes, lower byte first. This indicate there are 6 bytes of data from Node ID to XOR.
00 00 = Node ID, Serial number of CR038, 2 bytes, lower byte first. 00 00 mean broadcast, it works for any ID.
01 01 = Function/Command Code for Initialize Port, 2 bytes, lower byte first.

Baud = Parameter for Baud rate. 1 byte.

- 0 = 4800 bps
- 1 = 9600 bps
- 2 = 14400 bps
- 3 = 19200 bps
- 4 = 28800 bps
- 5 = 38400 bps
- 6 = 57600 bps
- 7 = 115200 bps

XOR = result of exclusive OR operation from Node ID to Baud, in this example.

Example: To set the UART Baud rate to 19200. You will need to send command in the current baud rate, not the desire baud rate. This is the command from host to CR038:

Command (Hexadecimal): AA BB 06 00 00 00 01 01 03 03

Reply from CR038 (Hexadecimal): AA BB 06 00 BF FF 01 01 00 40
BF FF = Node ID or Serial number of CR038, 2 bytes, lower byte 1st.
01 01 = Function/Command Code from host, 2 bytes, lower byte 1st.
00 = Status, 1 byte. 0 mean success, other than 0 mean fail.
40 = Result of XOR operation from Node ID to Status. Try it and you will get 0x40

Back to Function/Command Set

Set LED Status: 0x0107
Function: Set LED status on CR038

Packet Format (Hexadecimal): AA BB 06 00 00 00 07 01 LED XOR
AA BB = Header, 2 bytes.
06 00 = Packet length, 2 bytes, lower byte first. This indicate there are 6 bytes of data from Node ID to XOR.
00 00 = Node ID, Serial number of CR038, 2 bytes, lower byte first. 00 00 mean broadcast, it works for any ID.
07 01 = Function/Command Code Set LED status, 2 bytes, lower byte first.

LED = Parameter for LED status. 1 byte.

- 0 = Red LED OFF
- 1 to 3 = Red LED ON

XOR = result of exclusive OR operation from Node ID to LED, in this example.

Example: To Off the Red LED on CR038 (default is ON). This is the command from host to CR038:

Command (Hexadecimal): AA BB 06 00 00 00 07 01 00 06

Reply from CR038 (Hexadecimal): AA BB 06 00 BF FF 07 01 00 46
BF FF = Node ID or Serial number of CR038, 2 bytes, lower byte 1st. It can be any value.
07 01 = Function/Command Code from host, 2 bytes, lower byte 1st.
00 = Status, 1 byte. 0 mean success, other than 0 mean fail.
46 = Result of XOR operation from Node ID to Status.

The red LED will off after this command is sent.

Back to Function/Command Set

Set Antenna Status: 0x010C
Function: Set Antenna status on CR038, needed to access (read or write) to Mifare card.

Packet Format (Hexadecimal): AA BB 06 00 00 00 0C 01 Antenna XOR
AA BB = Header, 2 bytes.
06 00 = Packet length, 2 bytes, lower byte first. This indicate there are 6 bytes of data from Node ID to XOR.
00 00 = Node ID, Serial number of CR038, 2 bytes, lower byte first. 00 00 mean broadcast, it works for any ID.
0C 01 = Function/Command Code for Set Antenna Status, 2 bytes, lower byte first.

Antenna = Parameter for Antenna status. 1 byte.

- 0 = Antenna OFF
- 1 = Antenna ON

XOR = result of exclusive OR operation from Node ID to Antenna, in this example.

Example: To ON (Activate) Antenna on CR038 (default is OFF). This is the command from host to CR038:

Command (Hexadecimal): AA BB 06 00 00 00 0C 01 01 0C

Reply from CR038 (Hexadecimal): AA BB 06 00 BF FF 0C 01 00 4D
BF FF = Node ID or Serial number of CR038, 2 bytes, lower byte 1st. It can be any value.
0C 01 = Function/Command Code from host, 2 bytes, lower byte 1st.
00 = Status, 1 byte. 0 mean success, other than 0 mean fail.
4D = Result of XOR operation from Node ID to Status.

The Antenna on CR038 will be activated and you can further send command to access the Mifare card.

Back to Function/Command Set

Mifare Card Type Request: 0x0201
Function: Read Mifare Card type, the Mifare card must be near to CR038 (best is on top) and Antenna must be activated. This step is needed if you want to access the Mifare card.

Packet Format (Hexadecimal): AA BB 06 00 00 00 01 02 Request XOR
AA BB = Header, 2 bytes.
06 00 = Packet length, 2 bytes, lower byte first. This indicate there are 6 bytes of data from Node ID to XOR.
00 00 = Node ID, Serial number of CR038, 2 bytes, lower byte first. 00 00 mean broadcast, it works for any ID.
01 02 = Function/Command Code to request Mifare card type, 2 bytes, lower byte first.

Request = Request Mifare Card type code. 1 byte.

- 0x52 = Request all Type A card in the reading range
- 0x26 = Request all idle card

XOR = result of exclusive OR operation from Node ID to Request, in this example.

Example: To request available Mifare card type.

Command (Hexadecimal): AA BB 06 00 00 00 01 02 52 51

Reply from CR038 (Hexadecimal): AA BB 08 00 BF FF 01 02 00 04 00 47
BF FF = Node ID or Serial number of CR038, 2 bytes, lower byte 1st. It can be any value.
01 02 = Function/Command Code from host, 2 bytes, lower byte 1st.
00 = Status, 1 byte. 0 mean success, other than 0 mean fail.
04 00 = Mifare Card type: (lower byte 1st)

- 0x0044 = ultra light
- 0x0040 = Mifare_one(S50) Classic 1K
- 0x0020 = Mifare_One(S70)
- 0x0344 = Mifare_DESFire
- 0x0080 = Mifare_Pro
- 0x0304 = Mifare_ProX

47 = Result of XOR operation from Node ID to Status.

The type of Mifare card/tag/transponder will be returned and the cards will wake up. This step is needed before you can access to Mifare card/tag/transponder.

Back to Function/Command Set

Mifare Anti-collision: 0x0202
Function: Read the NUID of Mifare card/tag/transponder. The Mifare card must be near to CR038 (best is on top). This step is needed if you want to access the Mifare card.

Packet Format (Hexadecimal): AA BB 05 00 00 00 02 02 00
AA BB = Header, 2 bytes.
05 00 = Packet length, 2 bytes, lower byte first. This indicate there are 5 bytes of data from Node ID to XOR.
00 00 = Node ID, Serial number of CR038, 2 bytes, lower byte first. 00 00 mean broadcast, it works for any ID.
02 02 = Function/Command Code to perform anti-collision. It is to read the NUID from Mifare card, 2 bytes, lower byte first.
00 = result of exclusive OR operation from Node ID to Function/Command code, in this example.

Example: To request available Mifare card NUID.

Command (Hexadecimal): AA BB 05 00 00 00 05 02 00

Reply from CR038 (Hexadecimal): AA BB 0A 00 BF FF 02 02 00 46 FF A6 B8 E7
BF FF = Node ID or Serial number of CR038, 2 bytes, lower byte 1st. It can be any value.
02 02 = Function/Command Code from host, 2 bytes, lower byte 1st.
00 = Status, 1 byte. 0 mean success, other than 0 mean fail.
46 FF A6 B8 = Mifare Card NUID: (lower byte 1st), please refer to Manufacturer block, this is the data from byte 0 to byte 3.
E7 = Result of XOR operation from Node ID to NUID.

The NUID of Mifare card/tag/transponder will be returned and this ID is needed to select a particular Mifare to access it.

Back to Function/Command Set

Mifare Select Card: 0x0203

Function: Select a particular Mifare Card/Tag with the NUID. The Mifare card must be near to CR038 (best is on top). This step is needed if you want to access the Mifare card.

Packet Format (Hexadecimal): AA BB 09 00 00 00 03 02 XX XX XX XX XOR
AA BB = Header, 2 bytes.
09 00 = Packet length, 2 bytes, lower byte first. This indicate there are 9 bytes of data from Node ID to XOR.
00 00 = Node ID, Serial number of CR038, 2 bytes, lower byte first. 00 00 mean broadcast, it works for any ID.
03 02 = Function/Command Code to select particular with NUID. It will activate the particular Mifare card, 2 bytes, lower byte first.
XX XX XX XX = 4 bytes of NUID you obtain from Mifare Anti-collision, lower byte 1st.
XOR = result of exclusive OR operation from Node ID to NUID, in this example.

Example: To select a Mifare card with NUID
We take the NUID from the previous command: 46 FF A6 B8, the command should be (Hexadecimal):
AA BB 09 00 00 00 03 02 46 FF A6 B8 A6

Reply from CR038 (Hexadecimal): AA BB 07 00 BF FF 03 02 00 08 0A
BF FF = Node ID or Serial number of CR038, 2 bytes, lower byte 1st. It can be any value.
03 02 = Function/Command Code from host, 2 bytes, lower byte 1st.
00 = Status, 1 byte. 0 mean success, other than 0 mean fail.
08 = Select Acknowledge Code (SAK), 0x08 indicate is Mifare Classic 1K
0A = Result of XOR operation from Node ID to SAK.

Now, only that particular Mifare with NUID is activated and all further access will tied to this card.

Back to Function/Command Set

Mifare Halt: 0x0204

Function: To place the selected Mifare Card/Tag in halt mode, to deactivate the card. Once the card is halted, you will need to start from Mifare Request again to activate the card. The Mifare card must be near to CR038 (best is on top).

Packet Format (Hexadecimal): AA BB 05 00 00 00 04 02 06
AA BB = Header, 2 bytes.
05 00 = Packet length, 2 bytes, lower byte first. This indicate there are 5 bytes of data from Node ID to XOR.
00 00 = Node ID, Serial number of CR038, 2 bytes, lower byte first. 00 00 mean broadcast, it works for any ID.
04 02 = Function/Command Code to perform halt (deactivate) on selected card, 2 bytes, lower byte first.
06 = result of exclusive OR operation from Node ID to Function/Command code, in this example.

Example: To halt a selected Mifare card.

Command (Hexadecimal): AA BB 05 00 00 00 04 02 06

Reply from CR038 (Hexadecimal): AA BB 06 00 BF FF 04 02 00 46
BF FF = Node ID or Serial number of CR038, 2 bytes, lower byte 1st. It can be any value.
04 02 = Function/Command Code from host, 2 bytes, lower byte 1st.
00 = Status, 1 byte. 0 mean success, other than 0 mean fail.
0A = Result of XOR operation from Node ID to Status.

Now the selected Mifare card is deactivated and to reactivate, you will need to start the command from Mifare Request.

Back to Function/Command Set

Mifare Authentication2: 0x0207

Function: To perform authentication with selected Mifare card for memory access. You can select secret Key A or B as authentication password. If a fresh new card from factory, Key A is use for authentication purpose and is hexadecimal FF, FF, FF, FF, FF, FF (6 bytes of 0xFF). You will need to select the particular block for this purpose. It will authenticate for whole sector. Example if you choose the block to be 10, you are actually authenticate for whole sector 2. Access to block 8, 9, 10 and 11(block 11 is sector trailer) will be allowed if the authentication is successful. The selected Mifare card must be near to CR038 (best is on top). There are several steps before you can do authentication, please check the flow.

Packet Format (Hexadecimal): AA BB 0D 00 00 00 07 02 Mode Block Key XOR
AA BB = Header, 2 bytes.
0D 00 = Packet length, 2 bytes, lower byte first. This indicate there are 13 bytes of data from Node ID to XOR.
00 00 = Node ID, Serial number of CR038, 2 bytes, lower byte first. 00 00 mean broadcast, it works for any ID.
07 02 = Function/Command Code to perform authentication process with selected card, 2 bytes, lower byte first.
Mode = Authentication mode, to select using Key A or Key B for authentication purpose. 0x60 is to use Key A, 0x61 is to use Key B.
Block = Authentication block, with the same sector, it will match the sector trailer.
Key = Authentication Key, 6 bytes, lower byte 1st. This 6 bytes key must match the Key (A or B) in the sector trailer on the sector chosen.
XOR = result of exclusive OR operation from Node ID to NUID, in this example.

Example: To authenticate sector 3 with Key A. (default Key A is all 0xFF, 6 bytes)

Command (Hexadecimal): AA BB 0D 00 00 00 07 02 60 0A FF FF FF FF FF FF 6F

Reply from CR038 (Hexadecimal): AA BB 06 00 BF FF 07 02 00 45
BF FF = Node ID or Serial number of CR038, 2 bytes, lower byte 1st. It can be any value.
07 02 = Function/Command Code from host, 2 bytes, lower byte 1st.
00 = Status, 1 byte. 0 mean success, other than 0 mean fail.
45 = Result of XOR operation from Node ID to Status.

Now the authentication to selected Mifare card is successful and you can access to the blocks in the same sector. In this example, block 8, 9, 10 and 11 are in the same sector. BTW, the access bits do control the mode of access.

Back to Function/Command Set

Mifare Read: 0x0208

Function: To read the data of certain block (16 bytes) within a sector in the selected Mifare card, authentication must be successful. The selected Mifare card must be near to CR038 (best is on top). There are several steps before you can do authentication, please check the flow.

Packet Format (Hexadecimal): AA BB 06 00 00 00 08 02 Block XOR
AA BB = Header, 2 bytes.
06 00 = Packet length, 2 bytes, lower byte first. This indicate there are 6 bytes of data from Node ID to XOR.
00 00 = Node ID, Serial number of CR038, 2 bytes, lower byte first. 00 00 mean broadcast, it works for any ID.
08 02 = Function/Command Code to read a block of data from selected card, 2 bytes, lower byte first.
Block = Data block which you want to read.
XOR = result of exclusive OR operation from Node ID to Block, in this example.

Example: To read data in block 10.

Command (Hexadecimal): AA BB 06 00 00 00 08 02 0A 00

Reply from CR038 (Hexadecimal): AA BB 16 00 BF FF 08 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 4A
BF FF = Node ID or Serial number of CR038, 2 bytes, lower byte 1st. It can be any value.
08 02 = Function/Command Code from host, 2 bytes, lower byte 1st.
00 = Status, 1 byte. 0 mean success, other than 0 mean fail.
16 bytes of data = Data return from the read process, block 10. Factory default is all 0x00.
4A = Result of XOR operation from Node ID to Data.

Back to Function/Command Set

Mifare Write: 0x0209

Function: To write data into certain block (16 bytes) within a sector in the selected Mifare card, authentication must be successful. The selected Mifare card must be near to CR038 (best is on top). There are several steps before you can do authentication, please check the flow.

Packet Format (Hexadecimal): AA BB 16 00 00 00 09 02 Block D_0 D_1 D_2 D_3 D_4 D_5 D_6 D_7 D_8 D_9 D_A D_B D_C D_D D_E D_F XOR

AA BB = Header, 2 bytes.
16 00 = Packet length, 2 bytes, lower byte first. This indicate there are 22 bytes of data from Node ID to XOR.
00 00 = Node ID, Serial number of CR038, 2 bytes, lower byte first. 00 00 mean broadcast, it works for any ID.
09 02 = Function/Command Code to write data to a block of memory in selected card, 2 bytes, lower byte first.
Block = Data block which you want to read.
16 bytes of data to write = From D_0 to D_F. This is the 16 bytes of data needed for the CR038 to write into the Mifare card.
XOR = result of exclusive OR operation from Node ID to Data, in this example.

Example: To write data into block 10.

Command (Hexadecimal): AA BB 06 00 00 00 09 02 0A 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 01

Reply from CR038 (Hexadecimal): AA BB 16 00 BF FF 09 02 00 08
BF FF = Node ID or Serial number of CR038, 2 bytes, lower byte 1st. It can be any value.
09 02 = Function/Command Code from host, 2 bytes, lower byte 1st.
00 = Status, 1 byte. 0 mean success, other than 0 mean fail.
08 = Result of XOR operation from Node ID to Status.

Care should be taken when write data to sector trailer (block 3, 7, 11, 15, 19, 23, 27, 31, 35) because this block hold the Key A and Key B and most importantly, the Access bits. Wrong Access bits will lock the sector permanently.

Back to Function/Command Set

Mifare Initval: 0x020A

Function: To write value into certain value block within a sector in the selected Mifare card, authentication must be successful. The block must be in value format and the value is 4 bytes signed (two's complement) value. Mifare card must be near to CR038 (best is on top). There are several steps before you can do authentication, please check the flow.

Packet Format (Hexadecimal): AA BB 0A 00 00 00 0A 02 Block V_0 V_1 V_2 V_3 XOR

AA BB = Header, 2 bytes.
0A 00 = Packet length, 2 bytes, lower byte first. This indicate there are 10 bytes of data from Node ID to XOR.
00 00 = Node ID, Serial number of CR038, 2 bytes, lower byte first. 00 00 mean broadcast, it works for any ID.
0A 02 = Function/Command Code to write initial value to a value block of memory in selected card, 2 bytes, lower byte first.
Block = Value block where you want to write the initial value.
4 bytes of value to write = From V_0 to V_3. This is the 4 bytes of value in signed format needed for the CR038 to write into the Mifare card. Lower byte 1st.
XOR = result of exclusive OR operation from Node ID to value, in this example.

Example: To write initial value of 100 (decimal) into block 10.

Command (Hexadecimal): AA BB 0A 00 00 00 0A 02 0A 64 00 00 00 66

Reply from CR038 (Hexadecimal): AA BB 06 00 BF FF 0A 02 00 48
BF FF = Node ID or Serial number of CR038, 2 bytes, lower byte 1st. It can be any value.
0A 02 = Function/Command Code from host, 2 bytes, lower byte 1st.
00 = Status, 1 byte. 0 mean success, other than 0 mean fail.
48 = Result of XOR operation from Node ID to Status.

In this example, the block 10 will contain value of 100 (decimal) after this command.

Back to Function/Command Set

Mifare Read Balance: 0x020B

Function: To read the value balance from certain block (signed 4 bytes) within a sector in the selected Mifare card, authentication must be successful and the data stored in that block must be in value format. The selected Mifare card must be near to CR038 (best is on top). There are several steps before you can do authentication, please check the flow.

Packet Format (Hexadecimal): AA BB 06 00 00 00 0B 02 Block XOR

AA BB = Header, 2 bytes.
06 00 = Packet length, 2 bytes, lower byte first. This indicate there are 6 bytes of data from Node ID to XOR.
00 00 = Node ID, Serial number of CR038, 2 bytes, lower byte first. 00 00 mean broadcast, it works for any ID.
0B 02 = Function/Command Code to read the balance from a value block in selected card, 2 bytes, lower byte first.
Block = Data block which you want to read.
XOR = result of exclusive OR operation from Node ID to Block, in this example.

Example: To read the balance in value from block 10.

Command (Hexadecimal): AA BB 06 00 00 00 0B 02 0A 03

Reply from CR038 (Hexadecimal): AA BB 16 00 BF FF 0B 02 00 01 03 00 00 4B
BF FF = Node ID or Serial number of CR038, 2 bytes, lower byte 1st. It can be any value.
0B 02 = Function/Command Code from host, 2 bytes, lower byte 1st.
00 = Status, 1 byte. 0 mean success, other than 0 mean fail.
4 bytes of signed value = Balance of value in value block, lower significant value come first. it is stored in standard 2's complement format. In this example, 0x01, 0x03, 0x00, 0x00 represent 0x00000301 in actual case, so in decimal is 769.
4B = Result of XOR operation from Node ID to value.

Back to Function/Command Set

Mifare Decrement: 0x020C

Function: To subtract the value in a value block with a given value, further store the result in volatile memory. It will require 4 bytes of value for subtraction and block number. Authentication must be successful and the data stored in that block must be in value format. The selected Mifare card must be near to CR038 (best is on top). There are several steps before you can do authentication, please check the flow.

Packet Format (Hexadecimal): AA BB 0A 00 00 00 0C 02 Block V_0 V_1 V_2 V_3 XOR

AA BB = Header, 2 bytes.
0A 00 = Packet length, 2 bytes, lower byte first. This indicate there are 10 bytes of data from Node ID to XOR.
00 00 = Node ID, Serial number of CR038, 2 bytes, lower byte first. 00 00 mean broadcast, it works for any ID.
0C 02 = Function/Command Code to decrease value of V_0 to V_3 from value in the "Block", and store the result in volatile memory. 2 bytes, lower byte first.
Block = Value block where you want to subtract its value, this is the minuend.
4 bytes of subtrahend, lower byte come 1st and is signed value.
XOR = result of exclusive OR operation from Node ID to 4 bytes of subtrahend, in this example.

Example: To subtract 10 (decimal) from value in block 10.

Command (Hexadecimal): AA BB 0A 00 00 00 0C 02 0A 0A 00 00 00 0E

Reply from CR038 (Hexadecimal): AA BB 06 00 BF FF 0C 02 00 4E
BF FF = Node ID or Serial number of CR038, 2 bytes, lower byte 1st. It can be any value.
0C 02 = Function/Command Code from host, 2 bytes, lower byte 1st.
00 = Status, 1 byte. 0 mean success, other than 0 mean fail.
4E = Result of XOR operation from Node ID to Status.

In this example, if the initial value in block 10 is 201 (decimal) or 0x000000C9, it is will be subtract with 10 (decimal). Thus the result is 191 (decimal) or 0x000000BF. The result is stored in volatile memory, value in block does not changed.

Back to Function/Command Set

Mifare Increment: 0x020D

Function: To add the value in a value block with a given value, further store the result in volatile memory. It will require 4 bytes of value for addition and block number. Authentication must be successful and the data stored in that block must be in value format. The selected Mifare card must be near to CR038 (best is on top). There are several steps before you can do authentication, please check the flow.

Packet Format (Hexadecimal): AA BB 0A 00 00 00 0D 02 Block V_0 V_1 V_2 V_3 XOR

AA BB = Header, 2 bytes.
0A 00 = Packet length, 2 bytes, lower byte first. This indicate there are 10 bytes of data from Node ID to XOR.
00 00 = Node ID, Serial number of CR038, 2 bytes, lower byte first. 00 00 mean broadcast, it works for any ID.
0D 02 = Function/Command Code to add value of V_0 to V_3 with the value in the "Block", and store the result in volatile memory. 2 bytes, lower byte first.
Block = Value block where you want to add its value.
4 bytes of value for addition, lower byte come 1st and is signed value.
XOR = result of exclusive OR operation from Node ID to 4 bytes of value, in this example.

Example: To add 15 (decimal) into value in block 10.

Command (Hexadecimal): AA BB 0A 00 00 00 0D 02 0A 0F 00 00 00 0A

Reply from CR038 (Hexadecimal): AA BB 06 00 BF FF 0D 02 00 4F
BF FF = Node ID or Serial number of CR038, 2 bytes, lower byte 1st. It can be any value.
0D 02 = Function/Command Code from host, 2 bytes, lower byte 1st.
00 = Status, 1 byte. 0 mean success, other than 0 mean fail.
4F = Result of XOR operation from Node ID to Status.

In this example, if the initial value in block 10 is 206 (decimal) or 0x000000CE, it is will be added with 15 (decimal). Thus the result is 221 (decimal) or 0x000000DD. The result is stored in volatile memory, value in block does not changed.

Back to Function/Command Set

Mifare Restore: 0x020E

Function: To copy the value in a value block into volatile memory (RAM). Authentication must be successful and the data stored in that block must be in value format. The selected Mifare card must be near to CR038 (best is on top). There are several steps before you can do authentication, please check the flow.

Packet Format (Hexadecimal): AA BB 06 00 00 00 0E 02 Block XOR

AA BB = Header, 2 bytes.
06 00 = Packet length, 2 bytes, lower byte first. This indicate there are 6 bytes of data from Node ID to XOR.
00 00 = Node ID, Serial number of CR038, 2 bytes, lower byte first. 00 00 mean broadcast, it works for any ID.
0E 02 = Function/Command Code to copy value in a value block to volatile memory. 2 bytes, lower byte first.
Block = Value block where you want to copy its value.
XOR = result of exclusive OR operation from Node ID to Block, in this example.

Example: To copy value in block 10 to volatile memory.

Command (Hexadecimal): AA BB 06 00 00 00 0E 02 0A 06

Reply from CR038 (Hexadecimal): AA BB 06 00 BF FF 0E 02 00 4C
BF FF = Node ID or Serial number of CR038, 2 bytes, lower byte 1st. It can be any value.
0E 02 = Function/Command Code from host, 2 bytes, lower byte 1st.
00 = Status, 1 byte. 0 mean success, other than 0 mean fail.
4C = Result of XOR operation from Node ID to Status.

In this example, if the value in block 10 is 100 (decimal) or 0x00000064, it will be copied to volatile memory in Mifare card, value in block 10 does not changed.

Back to Function/Command Set

Mifare Transfer: 0x020F

Function: To copy the value in volatile memory into a value block. Authentication must be successful and the destination block must be in value format. The selected Mifare card must be near to CR038 (best is on top). There are several steps before you can do authentication, please check the flow.

Packet Format (Hexadecimal): AA BB 06 00 00 00 0F 02 Block XOR

AA BB = Header, 2 bytes.
06 00 = Packet length, 2 bytes, lower byte first. This indicate there are 6 bytes of data from Node ID to XOR.
00 00 = Node ID, Serial number of CR038, 2 bytes, lower byte first. 00 00 mean broadcast, it works for any ID.
0F 02 = Function/Command Code to copy value in volatile memory into a value block. 2 bytes, lower byte first.
Block = Destination value block where you the value in volatile memory to copy to.
XOR = result of exclusive OR operation from Node ID to Block, in this example.

Example: To copy value in volatile memory into block 10.

Command (Hexadecimal): AA BB 06 00 00 00 0F 02 0A 07

Reply from CR038 (Hexadecimal): AA BB 06 00 BF FF 0F 02 00 4D
BF FF = Node ID or Serial number of CR038, 2 bytes, lower byte 1st. It can be any value.
0F 02 = Function/Command Code from host, 2 bytes, lower byte 1st.
00 = Status, 1 byte. 0 mean success, other than 0 mean fail.
4D = Result of XOR operation from Node ID to Status.

In this example, if the value in volatile memory is 150 (decimal) or 0x00000096, it will be copied to block 10 and store in value format, value in block 10 change to 150 (decimal).

Back to Function/Command Set

7. WARRANTY

- Product warranty is valid for 6 months.
- Warranty only applies to manufacturing defect.
- Damaged caused by misuse is not covered under warranty
- Warranty does not cover freight cost for both ways.

Prepared by
Cytron Technologies Sdn. Bhd.
19, Jalan Kebudayaan 1A,
Taman Universiti,
81300 Skudai,
Johor, Malaysia.

Tel: +607-521 3178
Fax: +607-521 1861

URL: www.cytron.com.my
Email: support@cytron.com.my
sales@cytron.com.my

