

RHMMF1RW SERIAL COMMUNICATION PROTOCOL

1、INTRODUCTION

RHMMF1RW can read / write the Mifare I and its compatible card and uses serial port to communicate with host. The serial port can be connected to RS232 interface. Designer can connect RHMMF1RW directly to a PC's serial port without adding drivers. The host controls RHMMF1RW's operation by sending appropriate commands. The communication parameter is 9600bps, 1 start bit, 8 data bits and 1 stop bit without parity check bit. The least significant bit of one byte is transmitted first and the least significant byte of one communication data sequence is transmitted first.

2、CONTROL CHARACTER DEFINITION

Four control character have been defined:

Definition	Symbol	Value
Start Character	STX	0x02
End Character	ETX	0x03
Acknowledgement	ACK	0x06
Non-acknowledgement	NAK	0x15

3、COMMUNICATION PROCEDURE DESCRIPTION

A communication procedure is sponsored by the host sending commands and data to RHMMF1RW after a valid handshake. RHMMF1RW will return the result status and data after command execution.

The host send the command as the following:

HOST	DIRECTION	RHMMF1RW	COMMENT
STX	→		The host can resend STX if not receiving ACK or NAK in 20ms.
	←	ACK	
Command data Block + ETX	→		The host should start sending command data block in 45ms after receiving ACK. The interval between two consecutive byte in the command data block should be less than 15ms. During command data block sending, the synchronism is invalid if the host receive any data from RHMMF1RW and the host could stop command sending and restart the communication after 45ms.

The host starts a communication procedure by sending the STX to RHMMF1RW. The host can resend the STX or stop the communication if not receiving a ACK or NAK in 20ms. After receiving the ACK, the host sends the command data block containing command code and data. Then the host send ETX and waiting for the result.

RHMMF1RW will return the result in 300ms after receiving the host command.

RHMMF1RW	DIRECTION	HOST	COMMENT
STX	→		RHMMF1RW will stop sending if not receiving ACK in 45ms. RHMMF1RW will send out the result data block in 45ms after receiving the ACK.
	←	ACK	
Result Data Block + ETX	→		

RHMMF1RW sends STX and waits for the ACK from the host. RHMMF1RW will stop the communication if not receiving ACK in 45ms. RHMMF1RW will send out the result data block containing result status and data after receiving the ACK. Then RHMMF1RW sends ETX to complete the communication.

4、DATA BLOCK FORMAT

A. Command Data Block

<i>SeqNo</i>	<i>Cmd</i>	<i>Len</i>	<i>Data[]</i>	<i>BCC</i>
--------------	------------	------------	---------------	------------

SeqNo: 1 byte sequence number, value 0~255. RHMMF1RW will return the same sequence number in result data block. For simplicity, the *SeqNo* could be fixed to 0.

Cmd: 1 byte command code. 25 command codes are defined.

Len: 1 byte data length excluding *SeqNo*, *Cmd* and *Bcc*. Value 0~22. 0 for none.

Data[]: *Len* bytes command data. If *Len*=0, no this item.

BCC: 1 byte checksum defined as:

$$BCC = SeqNo \oplus Cmd \oplus Len \oplus Data[]$$

B. Result Data Block

<i>SeqNo</i>	<i>Status</i>	<i>Len</i>	<i>Data[]</i>	<i>BCC</i>
--------------	---------------	------------	---------------	------------

SeqNo: 1 byte sequence number.

Status: 1 byte result status. 0 for success and other value for error code.

Len: 1 byte data length excluding *SeqNo*, *Status* and *Bcc*. Value 0~16. 0 when any error occurred in command execution.

Data[]: *Len* bytes result data. If *Len*=0, no this item.

BCC: 1 byte checksum defined as:

$$BCC = SeqNo \oplus Cmd \oplus Len \oplus Data[]$$

5、COMMAND CODE LISTING

COMMAND		PARAMETER		COMMENT
NAME	VALUE	SEND	RECEIVE	
Request	0x41	_Mode	_TagType	Check if any card exist in inductive field.
Anticoll	0x42	Reserved	_Snr	Begin anti-collision procedure and return one card's UID.
Anticoll2	0x71	_Encoll, Reserved	_Snr	Return one card's UID with anti-collision feature enable or disabled
Select	0x43	_Snr	_Size	Select one card and return its capacity
Authentication	0x44	_Mode, Secnr	--	Authentication procedure
Authentication2	0x72	_Mode , _Secnr , _Keynr	--	Authentication with KEY in reader's EEPROM
AuthKey	0x73	_Mode , _Secnr , _Key,	--	Direct authentication
Halt	0x45	--	--	Halt card operation
Read	0x46	_Adr	_Data	Read 1 block(16 bytes) data
Write	0x47	_Adr, _Data	--	Write 1 block(16 bytes) data
Increment	0x48	_Adr, _Value	--	Increase the value of a value-block and save it in card's internal register.
Decrement	0x49	_Adr, _Value	--	Decrease the value of a value-block and save it in card's internal register.
Restore	0x4A	_Adr	--	Restore the content of one value-block to card's internal register
Transfer	0x4B	_Adr	--	Save the content of card's internal register to a value-block
Value	0x70	_Mode, _Adr, _Value _Trans, _Adr	--	Include increment, decrement, restore and transfer operation.
LoadKey	0x4C	_Mode , _Secnr , _nkey	--	Save KEY in reader's EEPROM
Reset	0x4E	_Msec	--	Close the inductive field for designated time
Set_LED_Bit	0x50	--	--	Turn off the green LED (RS232 version) or set the LED of reader to red(USB version)
Clr_LED_Bit	0x51	--	--	Turn on the green LED (RS232 version) or set the LED of reader to green(USB version)
Config	0x52	--	--	Reset and configure RHMMFIRW
Check_Write	0x53	_Snr, _Authmode, _Adr, _Data	--	Compare the content of a block with the transferred data
Buzzer	0x60	_Frequency , _Opentm, _Close tm, _Repcnt	--	Control the Buzzer beeping mode
Close	0x3F	--	--	Set RHMMFIRW to standby

Read_E2	0x61	_Adr, _Length	Data	Read EEPROM of the reader
Write_E2	0x62	_Adr, _Length, _Data	--	Write EEPROM of the reader

6. STATUS VALUE LISTING

NAME	VALUE	COMMENT
MI_OK, COMM_OK	0	Success
MI_NOTAGERR	1	No card in inductive field
MI_CRCERR	2	CRC error
MI_EMPTY	3	Value overflow
MI_AUTHERR	4	Authentication fail
MI_PARITYERR	5	Parity check fail
MI_CODEERR	6	BCC error
MI_SENDEERR	8	UID error
MI_KEYERR	9	KEY error
MI_NOTAUTHERR	10	Card not authenticated
MI_BITCOUNTERERR	11	Bit count error
MI_BYTECOUNTERERR	12	Byte count error
MI_TRANSERR	14	TANSNFER error
MI_WRITEERR	15	WRITE error
MI_INCRERR	16	INCREMENT error
MI_DECRERR	17	DECREMENT error
MI_READERR	18	READ error
MI_COLLERR	24	Collision error
MI_ACCESSTIMEOUT	27	Access time overflow
COMM_ERR	255	Serial communication error

7. COMMAND DESCRIPTION

7.1 Request

Send request command to detect if any card exist in inductive area.

HOST→RHMMF1RW		
Command	0x41	
Length	1	
Data[0]	_Mode	_Mode=0, all cards in area respond except those halted _Mode=1, all cards in area respond

RHMMF1RW→HOST		
Status	MI_OK, MI_QUIT, MI_NOTAGERR, MI_BITCOUNTERERR, COMM_ERR	
length	2	
Data[0]	_tagtype(LSB)	None when any error occurred
Data[1]	_tagtype(MSB)	None when any error occurred

7.2 Anticoll

Process anti-collision operation after request command.

HOST→RHMMF1RW		
Command	0x42	
length	1	
Data[0]	Reserved	Set to 0
RHMMF1RW→HOST		
status	MI_OK, MI_QUIT, MI_NOTAGERR, MI_BITCOUNTERERR, COMM_ERR	
length	4	
Data[0]	snr(LL)	Card's UID number
Data[1]	snr(LH)	
Data[2]	snr(HL)	
Data[3]	snr(HH)	

7.3 Select

Select one card with known UID and return its capacity.

HOST→RHMMF1RW		
Command:	0x43	
Length	4	
Data[0]	snr(LL)	Card's UID number
Data[1]	snr(LH)	
Data[2]	snr(HL)	
Data[3]	snr(HH)	

RHMMF1RW→HOST		
status	MI_OK, MI_QUIT, MI_NOTAGERR, MI_CRCERR, MI_PARITYERR, MI_BYTECOUNTERERR, COMM_ERR	
length	1	
Data[0]	_Size	Card's capacity

7.4 Authentication

Authenticate with KEY in the reader's EEPROM. The card should be authenticated before read, write and other data sector operation.

HOST→RHMMF1RW		
Command	0x44	
length	2	
Data[0]	_Mode	_Mode=0, authenticate with KEY A _Mode=1, authenticate with KEY B
Data[1]	_SecNr	The number of the sector(0~15) to be authenticated, with the KEY from the corresponding area in reader's EEPROM

RHMMF1RW→HOST		
Status	MI_OK, MI_QUIT, MI_NOTAGERR, AUTHERR, MI_BITCOUNTER, MI_PARITYERR, COMM_ERR	
Length	0	

7.5 Halt

Set the card in halted status.

HOST→RHMMF1RW		
Command	0x45	
length	0	

RHMMF1RW→HOST		
status	MI_OK, MI_QUIT, COMM_ERR	
length	0	

7.6 Read

Read out 1 block data from the card.

HOST→RHMMF1RW		
Command	0x46	
Length	1	
Data[0]	_Adr	Absolute block address

RHMMF1RW→HOST		
Status	MI_OK, MI_QUIT, MI_NOTAGERR, AUTHERR, MI_CRCERR, MI_NOTAUTHERR, MI_BITCOUNTER, MI_PARITYERR, COMM_ERR	
Length	16	
Data[0]	Data (0)	16 byte data
Data[1]	Data (1)	
...	...	
...	...	
Data[15]	Data (15)	

7.7 Write

Write 1 block data in the card

HOST→RHMMF1RW		
Command	0x47	
length	17	

Data[0]	_Address	Absolute block address
Data[1]		The first byte to be written
...		...
Data[16]		The last byte to be written
RHMMF1RW→HOST		
status	MI_OK, MI_QUIT, MI_NOTAGERR, AUTHERR, MI_BITCOUNTER, MI_WRITEERR, COMM_ERR	
length	0	

7.8 Increment

Read out the content of a value-block, check its structure and add it with the value assigned, then store the result in card's internal register. The value-block should have a specific format that is defined in Mifare series cards' datasheet.

HOST→RHMMF1RW		
Command	0x48	
Length	5	
Data[0]	_Adr	Absolute block address
Data[1]	_value(ll)	Value to be added
Data[2]	_value(lh)	
Data[3]	_value(hl)	
Data[4]	_value(hh)	

RHMMF1RW→HOST		
Status	MI_OK, MI_QUIT, MI_NOTAGERR, MI_NOTAUTHERR, MI_BITCOUNTER, MI_PARITYERR, COMM_ERR	
Length	0	

7.9 Decrement

Read out the content of a value-block, check its structure and subtract it with the value assigned, then store the result in card's internal register. The value-block should have a specific format that is defined in Mifare series cards' datasheet.

HOST→RHMMF1RW		
Command	0x49	
Length	5	
Data[0]	_Adr	Absolute block address
Data[1]	_value(ll)	Value to be subtracted
Data[2]	_value(lh)	
Data[3]	_value(hl)	
Data[4]	_value(hh)	

RHMMF1RW→HOST		
Status	MI_OK , MI_QUIT , MI_NOTAGERR , MI_NOTAUTHERR , MI_BITCOUNTERR , MI_PARITYERR, COMM_ERR	
length	0	

7.10 Restore

Read out the content of a value-block, check its structure and store it in card's internal register. The value-block should have a specific format which is defined in Mifare series cards' datasheet.

HOST→RHMMF1RW		
Command	0x4a	
Length	1	
Data[0]	_Adr	Absolute block address

RHMMF1RW→HOST		
Status	MI_OK , MI_QUIT , MI_NOTAGERR , MI_NOTAUTHERR , MI_BITCOUNTERR , MI_PARITYERR, COMM_ERR	
Length	0	

7.11 Transfer

Transfer the content of card's internal register to a valid value-block. The value-block should have a specific format which is defined in Mifare series cards' datasheet. This operation could only be used after increment, decrement or restore process.

HOST→RHMMF1RW		
Command	0x4b	
Length	1	
Data[0]	_Adr	Absolute block address

RHMMF1RW→HOST		
Status	MI_OK , MI_QUIT , MI_NOTAGERR , MI_NOTAUTHERR , MI_BITCOUNTERR , MI_PARITYERR, COMM_ERR	
Length	0	

7.12 Load_Key

Save a KEY in the reader's EEPROM. A config or close+config operation should be executed to make the save operation effective.

HOST→RHMMF1RW		
Command	0x4c	
Length	8	
Data[0]	_Mode	_Mode=0: KEY A _Mode=1: KEY B
Data[1]	_Secnr	Sector number in EEPROM and should be less than 16
Data[2]	_nkey[0]	6 bytes KEY with least significant byte first
...	...	
Data[7]	_nkey[5]	
RHMMF1RW→HOST		
status	MI_OK, MI_QUIT, MI_PARITYERR, COMM_ERR	
length	0	

7.13 Reset

Shut down the inductive field for a dictated time.

Example:

_Msec=0 : maintain in shut down status

_Msec=1 : shut down for 1ms

...

_Msec=0xff: shut down for 255ms

HOST→RHMMF1RW		
Command	0x4e	
length	1	
Data[0]	_Msec	Inductive field shut down duration (in millisecond unit)

RHMMF1RW→HOST		
status	MI_OK, MI_QUIT, COMM_ERR	
length	0	

7.15 Set_LED_Bit

Turn off the green LED (RS232 interface reader)

Set the LED in red (USB interface reader)

HOST→RHMMF1RW		
Command	0x50	
Length	0	

RHMMF1RW→HOST		
Status	MI_OK, MI_QUIT, COMM_ERR	
Length	0	

7.16 Clr_LED_Bit

Turn on the green LED (RS232 interface reader)

Set the LED in green (USB interface reader)

HOST→RHMMF1RW		
Command	0x51	
Length	0	

RHMMF1RW→HOST		
Status	MI_OK, MI_QUIT, COMM_ERR	
Length	0	

7.17 Config

RHMMF1RW should be configured after power on before any further process.

HOST→RHMMF1RW		
Command	0x52	
Length	0	

RHMMF1RW→HOST		
Status	MI_OK, MI_QUIT, COMM_ERR	
Length	0	

7.18 Check_Write

Compare the data written in the card with known data and return MIS_CHECK_OK if matched and MIS_CHK_COMPERR is mismatched. In this operation, RHMMF1RW use the KEY in its EEPROM with the same sector number of the card data block to be checked.

HOST→RHMMF1RW		
Command	0x53	
Length	22	
Data[0]	_Snr(ll)	Card's UID number
Data[1]	_Snr(lh)	
Data[2]	_Snr(hl)	
Data[3]	_Snr(hh)	
Data[4]	_Authode	Authentication mode used in last operation
Data[5]	_Adr	Absolute block address

Data[6]	Data(0)	16 data bytes for checking
⋮	⋮	
⋮	⋮	
Data[20]	Data(21)	

RHMMF1RW→HOST		
Status	MI_OK, MI_QUIT, MIS_CHECK_OK, MIS_CHK_FAILED, COMM_ERR, MIS_CHK_COMPERR	
Length	0	

7.19 Buzzer

Drive the BUZZER with dictated mode.

HOST→RHMMF1RW		
Command	0x60	
Length	4	
Data[0]	_Frequence	Reserved, fixed to 0
Data[1]	_Opentm	BUZZER active duration with 15ms resolution
Data[2]	_Closetm	BUZZER inactive duration with 15ms resolution
Data[3]	_Repcnt	Repeat times

RHMMF1RW→HOST		
Status	MI_OK, MI_QUIT, COMM_ERR	
Length	0	

7.20 READ_E2

Read out the content of the reader's EEPROM. The address should be less than 0x80. The content of the EEPROM with the address over 0x80 is used for KEY storage and could not be read out.

HOST→RHMMF1RW		
Command	0x61	
Length	2	
Data[0]	_Adr	Begin address of the EEPROM to be read out (less than 0x80).
Data[1]	_Length	Byte length of the data to be read out (less than 20 bytes).

RHMMF1RW→HOST		
Status	MI_OK, MI_QUIT, COMM_ERR	
Length	_Length	
Data[0]	Byte (0)	Data being read out
⋮	⋮	
⋮	⋮	
Data[_Length-1]	Byte (_Length-1)	

7.21 WRITE_E2

Write data in the reader's EEPROM. Data in address 0x00~0x0f are read-only product information. Data in address 0x10~0x2f are initialization data and should not be altered. Address area 0x80~0xff are used for KEY storage and could be written with Load_key command.

HOST→RHMMF1RW		
Command	0x62	
length	_Length +2	
Data[0]	_Adr	Begin address of the EEPROM to be written (0x30-0x7e)
Data[1]	_Length	Byte length of the data to be written (less than 20 bytes).
Data[2]	Data(0)	Data to be written
Data[length+1]	Data(Length)	

RHMMF1RW→HOST		
Status	MI_OK, MI_QUIT, COMM_ERR	
Length	0	

7.22 VALUE

Perform value-block related operation between one value block and one transfer block. The two block should both have value-block format and in the same sector. The value-block related operation includes increment, decrement and restore. Auto transfer is supported.

HOST→RHMMF1RW		
Command	0x70	
Length	7	
Data[0]	_Mode	_Mode=0xc0, decrement _Mode=0xc1, increment _Mode=0xc2, restore
Data[1]	_Adr	Absolute block address
Data[2]	_value(ll)	Value when in decrement or increment. No this item in restore operation.
Data[3]	_value(lh)	
Data[4]	_value(hl)	
Data[5]	_value(hh)	
Data[6]	_Trans_Adr	Absolute block address of the transfer block

RHMMF1RW→HOST		
Status	MI_OK, MI_QUIT, COMM_ERR, MI_NOTAERR, MI_BITCOUNTERR, MI_TRANSERR	
Length	0	

7.23 Close

Set the RHMMF1RW in Standby mode. Use config command to resume it active.

HOST→RHMMF1RW		
Command	0x3f	
length	0	

RHMMF1RW→HOST		
status	MI_OK, MI_QUIT, COMM_ERR	
length	0	

7.24 Anticoll2

Anti-collision procedure with multi-card enabled or disabled.

HOST→RHMMF1RW		
Command	0x71	
length	2	
Data[0]	_Encoll	If 1, multi-card is enabled. One card's UID number is returned after anti-collision. If 0, multi-card is disabled. An error is returned if multiple card detected.
Data[1]	Reserve	Reserve=0

RHMMF1RW→HOST		
status	MI_OK, MI_QUIT, COMM_ERR, MI_NOTAGERR, MI_COLLERR, MI_BITCOUNTERR	
length	4	
Data[0]	Snr(LL)	
Data[1]	Snr(LH)	
Data[2]	Snr(HL)	
Data[3]	Snr(HH)	

7.25 Authentication2

Cross-authenticate with KEY in reader's EEPROM. The card should be authenticated before read, write and other data sector operation.

HOST→RHMMF1RW		
Command	0x72	
Length	3	
Data[0]	_Mode	_Mode=0, authenticate with KEY A _Mode=1, authenticate with KEY B
Data[1]	_SecNr	The number of the sector to be authenticated
Data[2]	_KeyNr	The number of the sector in EEPROM storing the KEY

RHMMFIRW→HOST		
Status	MI_OK, MI_QUIT, MI_NOTAGERR, MI_PAROTERR, MI_PAROTUERR, COMM_ERR	
Length	0	

7.26 AuthKey

Directly authenticate with dictated KEY. The card should be authenticated before read, write and other data sector operation.

HOST→RHMMFIRW		
Command	0x73	
Length	8	
Data[0]	_Mode	_Mode=0, authenticate with KEY A _Mode=1, authenticate with KEY B
Data[1]	_SecNr	The number of the sector to be authenticated
Data[2]	_key(0)	6 bytes KEY
:	:	
:	:	
Data[7]	_key(5)	

RHMMFIRW→HOST		
status	MI_OK, MI_QUIT, MI_NOTAGERR, MI_PAROTERR, MI_PAROTUERR, COMM_ERR	
length	0	