USB HID Keyboard Simulator IC Card Reader

General Technical Manual

（Revision 3.04）

Beijing Jinmuyu Electronics Co., Ltd

2023/11/21

Contents

1 Introduction

In the systems of Windows, Linux and other PC systems support USB keyboard, the Reader simulate the USB keyboard to output the Card data. We supply configuration software. Under the software, user could configure complex read card method to implement much type of uses.

The reader supports the ISO14443 and ISO15693 compliant tags. The output data could be UID and/or Card data. And the output mode could be ASCII, Decimal, or Hexadecimal. For more configuration details, please refer to the following datasheet.

We supply many configuration templates. Uses could modify the template to fit the demands. And we will help users to make the configuration to resolve all the problems. Contact us is the most important thing.

2 Product Models

MR762
MR762B
MR7805
QM-ABCM7

3 Reader Installation

The Reader interface is USB HID class. The USB interface of reader is a keyboard simulator. The driver installation process is same to install a USB keyboard. The OS will automatic process the driver installation. What to do of users is plug USB header of reader to PC and wait.

After driver installation, the Host will show two Devices. One is HID Keyboard Device use for data output; another is HID-compliant device use for configuration.

4 Communication Protocol

4.1 Communication Format

4.1.1 Data packet sending format

- Host send:

Length	C.A.	Command	Data	Checksum

- Length: 2 bytes, number of bytes from Length byte to the last byte of Data, MSB first, length from 0x0004 to 0x01FE.

- C.A. (communication address): Not enabled, default 0x00.
- Command: 1 byte, the application layer command of the communication protocol, see 4.2 Command List for details
- Data: length depends on the command type, length from 0 to 506 bytes; depending on the processor, and some models will be less than 506 bytes.
- Checksum: 1 byte, Exclusive OR (XOR) results from length byte to the last byte of data.

4.1.2 Data returned format

- Success:

Length	C.A.	Command	Data	Checksum

- Failure:

Length	C.A.	Invert Command	Checksum

NOTE: "Failure" means that the communication between module and card failed.

4.2 List of commands

Commands	Meaning
0xE0	Control device
0xE1	Read device parameters
0xE2	Set device parameters
0xE3	Erase script
0xE4	Write script
0xE5	Enable script
0xF1	Read device version information

4.3 Command Details

4.3.1 Control device

Control LED. Note: Swiping the card will cause the LED light status to change.

Host send:

Length	0xE0	DATA	Checksum

DATA: 1byte

	Bit	Meaning
DATA[0]	Bit5~2	RFU
	Bit1	1=ON; 0=OFF; GREEN LED;
	Bit0	1=ON; 0=OFF; RED LED;

Success:

Length	0xE0	Checksum

Failure:

Length	0x1F	Checksum

4.3.2 Read Device Parameters

Read Device Parameters.

Host send:

Length	0xE1	Checksum

Success:

Length	0xE1	DATA	Checksum

DATA: 3bytes.

	Bit	Meaning
DATA[0]		Key rate, default 0x10, unit 100uS (not strict time)
DATA[1]	Bit7~2	RFU
	Bit1	0: 2nd set of scripts erased (default). 1: The second set of scripts has been written.
	Bit0	0: 1st set of scripts erased (default). 1: The first set of scripts has been written.
DATA[2]	Bit7~1	RFU
	Bit0	0: The first set of scripts is the current script; (default). 1: The second set of scripts is the current script.

Note: MR762 does not support the second set of scripts, DATA [1] and DATA [2] are meaningless.

Failure:

Length	0x1E	Checksum

4.3.3 Setting Device Parameters

To Set Device Parameters

Host send:

Length	0xE2	DATA	Checksum

DATA: 3bytes;

	Bit	Meaning
DATA[0]		Key rate, default 0x10, unit 1mS (not strict time).
DATA[1]		Script status, read-only, ignores current settings.

DATA[2]	Bit7~1	RFU
	Bit0	0: The first set of scripts is the current script; (default). 1: The second set of scripts is the current script.

Note: MR762 does not support the second set of scripts, DATA[1] and DATA[2] are meaningless.

Success:

Length	0xE2	Checksum

Failure:

Length	0x1D	Checksum

4.3.4 Erase script

Erase the 1st or 2nd set of script data according to the device parameters. It must be wiped every time, when the reader script is reconfigured. After the script is erased, the card reader does not work properly until the script is written.

Host send:

Length	0xE3	0xA5	0x01	Checksum

Success:

Length	0xE3	Checksum

Failure:

Length	0x1C	Checksum

4.3.5 Write script

Write the 1st or 2nd set of script commands according to the device parameters. The write script command is a repeated command. Every time you write, the device automatically saves; if you write incorrectly, or you want to rewrite, you need to erase and restart the script sequence.

Host send:

Length	0xE4	DATA	Checksum

DATA: n bytes, see Chapter 5 Script Commands for details.

Success:

Length	0xE4	Checksum

Failure:

Length	0x1B	Checksum

4.3.6 Enable script

Scripts can be enabled after writing the script, and without rebooting the device.

Host send:

Length	0xE5	0xA5	0x01	Checksum

Success:

Length	0xE5	Checksum

Failure:

Length	0x1A	Checksum

4.3.7 Read device version information

Read the device version information, the version information is encoded according to ASCII.

Host send:

Length	0xF1	Checksum

Success:

Length	0xF1	DATA	Checksum

DATA: n bytes, version information.

Failure:

Length	0x0E	Checksum

5 Script commands

5.1 List of commands

Commands	Meaning
0x1X	SAM Card operation
0x2X	CPU Card operation
0x3X	Output operation
0x4X	RAM operation
0x5X	MIFARE Card operation
0x6X	Ultralight Card operation
0x7X	SR series card
0x8X	ISO15693 Card operation

5.2 SAM Card operation

DATA[0]	DATA[1~31]
CMD	APDU

CMD： 0x11 Data (APDU) to be sent to SAM Card, Returned Data will be stored in RAM1.
0x12 Data (APDU) to be sent to SAM Card, Returned Data will be stored in RAM2.

APDU： COS command

5.3 CPU Card operation

DATA[0]	DATA[1~31]
CMD	APDU

CMD： 0x21 Data (APDU) to be sent to CPU Card, Returned Data will be stored in RAM1.
0x22 Data (APDU) to be sent to CPU Card, Returned Data will be stored in RAM2.
0x2A CPU-A card reset operation, no APDU

APDU： COS command

Example:

Meaning	**Script**	**Actual output**
CPU A card reset	Send: 2A Recv: 00	"The reset operation of the card will be completed"
Output reset information	Send: 34 22 00 00 Recv: 00	CPU-A card reset information： 1078809002209000000000002100B822
APDU read random number	Send: 21 00 84 00 00 08 Recv: 00	"The result will be stored in RAM1"
Output random number, which is the data in RAM1	Send: 31 22 00 0A Recv: 00	45478466732A8B899000

5.4 Output operation

DATA[0]	DATA[1]	DATA[2]	DATA[3]
CMD	Mode	Start/ KeyValue	Length

CMD： 0x30 data in the current command
0x31 RAM1 data output
0x32 RAM2 data output
0x33 SNR output
0x34 Card reset information output

Commands list:

<table>
<tr><th>DATA[0]</th><th colspan="3">DATA[1]</th><th>DATA[2]</th><th>DATA[3]</th></tr>
<tr><td rowspan="7">0x30：(data in the current command)
0x31: RAM1 output

0x32：RAM2 output

0x33：SNR output

0x34：(Card reset information output)</td><td>Bit7- Bit6-Bit5 -Bit4
(output method)</td><td colspan="2">Bit3- Bit2- Bit1- Bit0</td><td></td><td></td></tr>
<tr><td rowspan="4">0001:Decimal output, default leading zero output

1001:Decimal output, leading zeros are not output</td><td>Bit3</td><td>The data length and combination length are not integer multiples, and the zero-fill method for the last data.
0: Default to add zero behind it.
1: Add zeros in front.</td><td rowspan="5">Start：(start byte of output data)</td><td rowspan="5">Length: output byte length (use default length, this byte is 0x00).</td></tr>
<tr><td>Bit2</td><td rowspan="3">Conversion form:
001: Convert one byte to be decimal form;
010: Convert two bytes to be decimal form;
011: Convert three bytes to be decimal form;
100: Convert four bytes to be decimal form.</td></tr>
<tr><td>Bit1</td></tr>
<tr><td>Bit0</td></tr>
<tr><td rowspan="4">0010: Hex output</td><td>Bit3</td><td>Data forward and reverse order control;
0: Default positive order;
1: Reverse order;</td></tr>
<tr><td>Bit2</td><td>RFU</td></tr>
<tr><td>Bit1</td><td>Case control of letters in output data:
0: lowercase letters;
1: capital letters.</td></tr>
<tr><td>Bit0</td><td>RFU</td></tr>
<tr><td>0100: Directly output the data in RAM1 or RAM2</td><td>Bit3
~
Bit0</td><td>RFU</td><td>Start: the starting byte of output data</td><td>Length: Output byte length (use the default length, this byte is 0x00)</td></tr>
<tr><td rowspan="4">0011:keyvalue output</td><td>Bit3</td><td>RFU</td><td rowspan="4">KeyValue</td><td rowspan="4">none</td></tr>
<tr><td>Bit2</td><td>Alt key control:
0: not output;
1: output.</td></tr>
<tr><td>Bit1</td><td>Shift key control:
0: not output;
1: output.</td></tr>
<tr><td>Bit0</td><td>Ctrl key control:
0: not output;</td></tr>
</table>

			1: output.		
	0101: ASCII code format	Bit3	RFU	none	none
		Bit2	RFU		
		Bit1	RFU		
		Bit0	RFU		

Example:

Explanation	Script	Actual output
Output M1 card SNR; Hex; Uppercase	Send: 33 22 00 00 Recv: 00	831194DD
Output M1 card SNR; Decimal; Single byte conversion; The leading zero is not output.	Send: 33 91 00 00 Recv: 00	13117148221
Output M1 card SNR; Decimal; Double byte conversion.	Send: 33 12 00 00 Recv: 00	3355338109
Output M1 card SNR; Decimal; three-byte conversion.	Send: 33 13 00 00 Recv: 00	0858971614483456
Output M1 card SNR; Decimal; three-byte conversion; Start zero not output.	Send: 33 1B 00 00 Recv: 00	0858971600000221
Output M1 card SNR; Decimal; four-byte conversion.	Send: 33 14 00 00 Recv: 00	2198967517
Output RAM1; Hex uppercase	Send: 3122 00 04 Recv: 00	"RAM1, the current 4 bytes of data"
Output RAM2; Hex uppercase	Send: 32 22 00 10 Recv: 00	"RAM2, the current 10 bytes of data"
Decimal; output 12345678.	Send: 30 13 BC614E Recv: 00	12345678
Output enter	Send: 30 30 58 Recv: 00	"Enter "
Key-value output ! @ # $ % ^ & * ( ) _ + { } | : " ~ < > ?	Send: 30 32 1E 1F 20 21 22 23 24 25 26 27 2D 2E 2F 30 31 33 34 35 36 37 38 Recv: 00	! @ # $ % ^ & * ( ) _ + { } | : " ~ < > ?
Output the string "jinmuyudianziyouxiangongsi" in the current data when swiping the card.	Send: 30 50 6A 69 6E 6D 75 79 75 64 69 61 6E 7A 69 79 6F 75 78 69 61 6E 67 6F 6E 67 73 69 Recv: 00	jinmuyudianziyouxiangongsi

Note:

Take UltraLight card number as an example:

Convert the reading card number (or data) into decimal form every two bytes, and fill with zeros at the back end of the data if it is not neat.

Script command： 33 12 00 00

Datasource data： 04 23 BB E1 ED 25 80

Conversion data： 0423 BBE1 ED25 0080

Output Data： 01059 48097 60709 00128

The datasource data is obtained, that is, the card number is 0x04 23 bb e1 ed 25 80. The script command requires zero-padded data backend, so the converted data is 0x0423 bbe1 ed25 8000. The script command requires conversion to decimal every two bytes, and the maximum number of two bytes converted to decimal is 65536, which is five digits. Therefore, the corresponding decimal number of 0x0423 is converted to 01059, so the output data 01059 48097 60709 32768 is obtained by one conversion.

5.5 RAM Operation

DATA[0]	DATA[1]	DATA[2]	DATA[3]	DATA[4]	DATA[6~n]
Source	Operation method	Source Start Addr. / Data source location	Target Start Addr. / Operand value	Length /no	Data/no

Commands list:

<table>
<tr><th>DATA[0]</th><th colspan="3">DATA[1]</th><th>DATA[2]</th><th>DATA[3]</th><th>DATA[5]</th><th>DATA[6~n]</th></tr>
<tr><td>Data source</td><td colspan="3">Operation method</td><td></td><td></td><td></td><td></td></tr>
<tr><td rowspan="5">0x40：(The data in the current instruction is the data source).
0x41：RAM1 data is the data source.
0x42：RAM2 data is the data source
0x43：Card number is the data source.</td><td colspan="3">0x11：Copy data source to RAM1.
0x12：Copy data source to RAM2.</td><td>Data source starting position.</td><td>Target starting position.</td><td>Operation data length.</td><td>Data: When DATA[0]=0x40, this part is valid, otherwise it is empty.</td></tr>
<tr><td colspan="3">0x21：Addition operation.
0x22：Subtraction operation.</td><td>Specify the data source location.</td><td>Operand value: 1 byte, range 0x00~0xFF.</td><td>None</td><td>None</td></tr>
<tr><td colspan="3">0x31：BCD to HEX code.
0x32：HEX to BCD code.</td><td>Specify the data source location.</td><td>None</td><td>None</td><td>None</td></tr>
<tr><td>Bit7-Bit6-Bit5-Bit4</td><td colspan="2">Bit3-Bit2-Bit1-Bit0</td><td colspan="4"></td></tr>
<tr><td>0100: HEX data manipulation instructions</td><td>Bit3</td><td>Data forward and reverse order control:
0: Default positive order;
1: Reverse Order.</td><td rowspan="4">Specify the data source location.</td><td rowspan="4">Operation data length.</td><td rowspan="4">None</td><td rowspan="4">None</td></tr>
<tr><td></td><td>Bit2</td><td>Byte high and low bit swap control:
0: No swap by default;
1: swap.</td></tr>
<tr><td></td><td>Bit1</td><td>Byte high and low nibble swap control:
0: No swap by default;
1: swap.</td></tr>
<tr><td></td><td>Bit0</td><td>Byte bit wise</td></tr>
</table>

			inversion control :0: No reverse by default; 1: Reverse				
	0101： Decimal conversion operation	Bit2 ~ Bit0	Convert form: 001: Convert a byte to decimal form 010: Convert two bytes to decimal form 011: Convert three bytes to decimal form 100: Four bytes converted to decimal form 111: The default length is converted to decimal form	Specify data source location	Operation data length	none	none

Example:

Explanation	Script	Actual output
The data in the current instruction is the data source, copied to RAM1.	Send: 40 11 00 00 04 11 22 33 44 Recv: 00	"The data currently saved in RAM1 is 0x11 22 33 44"
The card number is the data source, copied to RAM1.	Send: 43 11 00 00 04 Recv: 00	"Copy 4 bytes card number to RAM1"
The data in RAM1 is the data source, copied to RAM2.	Send: 41 12 00 00 10 Recv: 00	"The 16 bytes of data in RAM2 are the same as the data in RAM1"
Add 0x02 to the first byte of the card number in RAM1.	Send: 41 21 00 02 Recv: 00	Original Card ID: 831194DD Card new ID: 851194DD
The 3rd byte of the card number in RAM1 minus 0x03.	Send: 41 22 02 03 Recv: 00	Original Card ID: 831194DD Card new ID: 851191DD
The 3rd byte of the card number in RAM1 is converted from BCD to HEX code.	Send: 41 31 02 Recv: 00	Original Card ID: 831194DD Card new ID: 830B94DD
The second byte of the card number in RAM1 is converted from HEX to BCD code.	Send: 41 32 01 Recv: 00	Original Card ID: 831194DD Card new ID: 831794DD
Reverse sequence control of card numbers in RAM1.	Send: 41 48 00 04 Recv: 00	Original Card ID: 831194DD Card new ID: DD941183
The high and low bits of the card number in RAM1 are swapped.	Send: 41 44 00 04 Recv: 00	Original Card ID: 831194DD Card new SNR: C18829BB
Card number high nibble and low nibble swap in RAM1.	Send: 41 42 00 04 Recv: 00	Original Card ID: 831194DD Card new ID: 381149DD
Bitwise inversion control of card number in RAM1.	Send: 41 41 00 04 Recv: 00	Original Card ID: 831194DD Card new ID: 7CEE6B22

5.6 MIFARE Card operation

DATA[0]	DATA[1]	DATA[2]	DATA[3~9]
Data storage location	Block address	Key type	6 bytes key

Data storage location：0x51 MIFARE card operation results are stored in RAM 1.
0x52 MIFARE card operation results are stored in RAM 2.

Key type： 0x60 Key A
0x61 Key B

Example:

Explanation	Script	Actual output
Read block 0 of MIFARE 1 card and store the result in RAM1.	Send: 51 00 60 FFFFFFFFFFFF Recv: 00	16 bytes of block 0 will be stored in RAM 1.
Read block 4 of MIFARE 1 card and store result in RAM2.	Send: 52 04 60 FFFFFFFFFFFF Recv: 00	16 bytes of block 4 will be stored in RAM 2.

// Read block 0

Send: 51 00 60 FFFFFFFFFFFF

Recv: 00

// Output 16 bytes of data

Send: 31 22 00 10

Recv: 00

5.7 UltraLight Card operation

DATA[0]	DATA[1]
Data storage location	Start block address

Data storage location：0x61 the operation result is stored in RAM1.

0x62 the operation result is stored in RAM2.

Start block address： To read the starting blocks address.

Example:

Explanation	**Script**	**Actual output**
Read block 4~7 of UltraLight card and store the result in RAM1.	Send: 61 04 Recv: 00	The 16 bytes of blocks 4~7 will be stored in RAM1.

// Read block 4

Send: 61 04

Recv: 00

// to output the 16bytes data from the starting block4

Send: 31 22 00 10

Recv: 00

5.8 SR series card operation

DATA[0]	DATA[1]
Data storage location	Block address

Data storage location：0x71 the operation result is stored in RAM1.

0x72 the operation result is stored in RAM2.

Block address： The block address of the data to be read

Example:

Explanation	**Script**	**Actual output**
Read block 10 of the SRIX4K card and store the result in RAM 1.	Send: 71 0A Recv: 00	4 bytes of block 10 will be stored in RAM 1.

// Read block 10

Send: 71 0A

Recv: 00

// to output the 4 bytes data from the block 10

Send: 31 22 00 04

Recv: 00

5.9 ISO15693 Card operation

DATA[0]	DATA[1]	DATA[2]
Data storage location	Block address	Number of blocks

Data storage location：0x81 the operation result is stored in RAM1.
0x82 the operation result is stored in RAM2.

Block address： Read the first address of the block

Number of blocks： To read the Number of blocks

Example:

Explanation	**Script**	**Actual output**
Read the block 1 of the ISO15693 card as the 8 data blocks of the starting block, and store the result in RAM1.	Send: 81 01 08 Recv: 00	8 blocks of data starting with block 1, each block is 4 bytes, a total of 32 bytes will be stored in RAM1.

// ISO15693 read block 1~8
Send: 81 01 07
Recv: 00
// Output data of blocks 1~8, each block is 4 bytes, a total of 32 bytes, that is, 0x20 bytes
Send: 31 22 00 20
Recv: 00

Appendix A Common key-value table

Common additional key-value table

Key Name	HID Usage ID
Enter	0x58
Backspace	0x2A
Tab	0x2B
Space	0x2C
Right Arrow	0x4F
Left Arrow	0x50
Down Arrow	0x51
Up Arrow	0x52

For other attachment keys, please refer to "USB HID to PS2 Scan Code Translation Table.pdf". Sorry for not testing all HID Usage IDs.

Documentation update record

Version	Date	Changes
V2.20		Initial version
V2.21	March 25, 2015	Modified overall table style; Added script examples; Adjust article format;
V2.31	Sept. 28, 2018	Update file content format; Added example instructions section;
V3.00	June 24, 2022	An introduction to adding communication protocols; Modify some description errors;
V3.01	Oct. 16, 2023	Add the support to MR7805 USB HID module
V3.02	Sept. 18, 2023	Add format type for decimal output
V3.03	Oct. 19, 2023	Add format type for hexadecimal output
V3.04	Nov. 21, 2023	Add HEX to DEC function in adding RAM operation; Add the ability to directly output the data in RAM1 or RAM2 in the output operation;