

# COMMUNICATION    PROTOCOL

---

## RFID reader with bluetooth function

**MODEL: HY-87D**

**REV: 1.1**

This document define exchange data communication protocol between the RFID reader with bluetooth function and the host(such as PC,PDA etc.)with bluetooth function

# Directory

<b>Bluetooth reader communication protocol.....</b>	<b>3</b>
<b>一、 Description.....</b>	<b>3</b>
<b>二、 Protocol specification .....</b>	<b>3</b>
<b>2.1 work mode.....</b>	<b>3</b>
<b>2.2Definition .....</b>	<b>4</b>
<b>2.3Data.....</b>	<b>4</b>
<b>三、 Protocol format .....</b>	<b>4</b>
<b>3.1 Command format.....</b>	<b>4</b>
<b>3.2 Response format .....</b>	<b>5</b>
<b>四、 Protocol explained .....</b>	<b>5</b>
<b>4.1Name explained .....</b>	<b>5</b>
<b>4.2Protocol analysis .....</b>	<b>5</b>

# Bluetooth Reader Communication protocol

## 一、Description

This bluetooth reader (in the following called reader) through bluetooth module provide serial port service and host with bluetooth function (in the following are called host ) exchange data , Reader as slave, baudrate : 9600 bps, data: 8 bits, parity: no, stop : 1.

This product supports ISO14443 type A protocol, such as MIFARE one tags.

## 二、Protocol specification

### 2.1 work mode

The reader has two modes selectable: 1. Key trigger mode; 2. Automatic detection mode (the two models can be setted by communication command, the default is: mode 1). Reader first bluetooth pairing (initial pairing password: 0000), and after a successful connection, by triggering the reader key (mode 1), the induction area within the tag data read, at the same time to produce a sound prompt, And the read data though bluetooth transmission to the host; or when the read is configured to "Automatic detection mode" (mode 2). Directly to the tag is close to the reader the induction area, data via the bluetooth upload to the host.

At the same time, the data recorded in a local memory, can be connected through the USB line to host.

### 2.2 Definition

**Frame :** A complete data format called a frame.

**Command frame:** The host send command to the reader (Such as check equipment ID, set date/clock, get current date/clock, erase the local record, read the local record etc.)

**Response frame:** The reader return data to the host. (include read tag's information through button trigger)

**Byte:** 8 bits to a byte.

## 2.3 Data

Host: the single frame data maximum to 127 bytes;

Reader: the single frame data maximum to 127 bytes;

## 三、 Protocol format

### 3.1 Command format

	head	length	command	data	check	end
bytes	1	1	1	N	1	1

### 3.2 Response format

	head	length	command	status	data	check	end
bytes	1	1	1	1	N	1	1

## 四、 Protocol explained

### 4.1 Name explained

**head:** 0x55;

**length:** The sum of all bytes, excluding head and end.

**Command:** Different function code (HEX format)

**data:** The parameters passed.

**Status:** Successful: 0x00; failure: other

**Check:** BCC(reference <BCC calculation sub-program>>)

**End:** 0xAA

## 4.2 Protocol analysis

*note: all are 16 hexadecimal character frame format*

### 1. Operation mode selection

Command frame:

head	length	command	data	check	end
0x55	0x04	0x40	0x01/0x00	0x10/0x11	0xAA

Explanation:

data : 0x01 ----- key trigger mode ,(mode 1, it's default mode)

0x00----- automatic detection mode,(mode 2)

Response frame:

head	length	command	status	data	check	end
0x55	0x04	0x40	0x00	none	0x11	0xAA

### 2. Buzzer set command

Command frame:

head	length	command	data	check	end
0x55	0x04	0x0C	0x01/0x02	0x5C/0x5F	0xAA

explanation:

data: 0x01--- buzzer turn on (default); 0x02----buzzer turn off.

Response frame:

head	length	command	status	data	check	end
0x55	0x04	0x0C	0x00	none	0x5D	0xAA

### 3. Date/Clock set command

Command frame:

head	length	command	data	check	end
0x55	0x0A	0x0D	XX...XX	XX	0xAA

explanation:

data: According to the second, minute, hour, day, month, week, year sequence, each representing one byte.

Response frame:

head	length	command	status	data	check	end
0x55	0x04	0x0D	0x00	none	0x5C	0xAA

**Example:**

**command:** 55   0A   0D   28 3B 17 1C 02 07 0B   44   AA  
                    ①   ②   ③                                      ④                                      ⑤   ⑥

In which: ①: head byte

②: length byte

③: date /clock set command code byte

④: second, minute, hour, day, month, week, year  
parameter(hex format) ,

⑤: BCC byte.

⑥: end byte

**response:**   55   04   0D   00   5C   AA  
                    ①   ②   ③   ④   ⑤   ⑥

In which: Ellipsis.

#### 4. Get Date/Clock command

Command frame:

head	length	command	data	check	end
0x55	0x03	0x0E	None	0x58	0xAA

Response frame:

head	length	command	status	data	check	end
0x55	0x0B	0x0E	0x00	XX..XX	XX	0xAA

explanation:

data: According to the second, minute, hour, day, month, year, week sequence, each representing one byte.

**Example:**

command: 55 03 0E 58 AA

① ② ③ ④ ⑤

In which: Ellipsis.

response: 55 0B 0E 00 46 45 05 01 03 11 07 42 AA

① ② ③ ④ ⑤ ⑥ ⑦

In which: ①: head byte

②: length byte

③: date /clock read command code byte

④: status byte

⑤: the current date/clock.(BCD code format

seconds,minutes,hour,days,months,years,week)

⑥: BCC byte.

⑦: end byte

## 5. Get machine ID command

Command frame:

head	length	command	data	check	end
0x55	0x03	0x0F	None	0x59	0xAA

Response frame:

head	length	command	status	data	check	end
0x55	0x0B	0x0F	0x00	XX..XX	XX	0xAA

**Example:**

**command:** 55 03 0F 59 AA

① ② ③ ④ ⑤

In which: Ellipsis.

**response:** 55 0B 0F 00 00 03 00 B5 01 30 80 56 AA

① ② ③ ④ ⑤ ⑥ ⑦

In which: ⑤: equipment ID number. (Different equipment ID not the same.).

## 6. Erase local record command

Command frame:



head	length	command	data	check	end
0x55	0x03	0x10	None	0x46	0xAA

Response frame:

head	length	command	status	data	check	end
0x55	0x04	0x10	0x00	none	0x41	0xAA

explanation:

In response to the frame each return again, will erase 6 records and forbidden to terminate the erase information.

Execute a command, should return 501 response frame information, we can erase all records.

## 7. Read local record command

Command frame:

head	length	command	data	check	end
0x55	0x03	0x11	None	0x47	0xAA

Response frame:

head	length	command	status	data	check	end
0x55	0x0E	0x11	0x00	XX...XX	XX	0xAA

explanation:

1. The data sequence UID ( accounted for 4 consecutive bytes ), record the time and date ( 6 bytes ); a total of 10 bytes.

2 . In response to the frame each return again, returns 1 records information. According to the stored time sequence of read, and must all records are read out, automatic stop (i.e., read over 3000 records information to stop responding. ).

## 8. Read UID (Through key trigger )

Command frame:

None.

Response frame:

head	length	command	status	data	check	end
0x55	0x0E	0x20	0x00	XX...XX	XX	0xAA

### explanation:

1. The data sequence UID ( accounted for 4 consecutive bytes ), record the time and date ( 6 bytes ); a total of 10 bytes.

**Example:** Each a trigger key, if read a tag, it returns:

55 0E 20 00 4C B7 EA D5 49 59 23 28 02 11 B7 AA

①

②

In which: ① the tag's UID, Different tags, the UID only.

② the current date/clock.

## 9. Request tag's UID command

Command frame:

head	length	command	data	check	end
0x55	0x04	0x51	0x52/0x26	0x52/0x26	0xAA

**explanation:**

data: 0x52-----find all the tags ; 0x26-----find idle tags.

Response frame:

head	length	command	status	data	check	end
0x55	0x06	0x51	0x00	XX...XX	XX	0xAA

**explanation:**

Data: XX XX-----returned by the card types, such as Mifare 1,

S50: 0X04, 0X00 ; S70 : 0X02, 0X00;

Low byte first, high byte last.

## 10. Anti collision command

Command frame:

head	length	command	data1	data2	check	end
0x55	0x04	0x52	0x93	0x00	0x91	0xAA

Response frame:

head	length	command	status	data	check	end
0x55	0x08	0x52	0x00	xx,xx,xx,xx	XX	0xAA

**explanation:**

**data: xx,xx,xx,xx-----tag's UID.**

**Example:**

**command:** 55 05 52 93 00 91 AA

**response:** 55 08 52 00 B2 82 C7 7D 85 AA

①

In which : ①: tag's UID, 4 bytes.low byte first,high byte last.

## 11. Select command

Command frame:

head	length	command	data1	data2	check	end
0x55	0x08	0x53	0x93	xx....xx	xx	0xAA

**explanation:**

**data2: Known tag's UID, 4 bytes.low byte first,high byte last.**

Response frame:

head	length	command	status	data	check	end
0x55	0x08	0x53	0x00	xx....xx	XX	0xAA

**explanation:**

**data: 4 bytes, first byte is 0x08,other bytes are part of the UID.**

## 12. Key authentication command

Command frame:

head	length	command	data1	data2	data3	data4	check	end
------	--------	---------	-------	-------	-------	-------	-------	-----

0x55	0x0F	0x54	0x60/ 0x61	xx....xx	xx...xx	xx	XX	0xAA
------	------	------	---------------	----------	---------	----	----	------

**explanation:**

data1: 0x60----- Key A. 0x61----Key B

data2: known tag's UID,

data3: key bytes, total 6 bytes.

data4: block number, Such as S50(0X00~0X3F),S70(0X00~0XFF).

Response frame:

head	length	command	status	data	check	end
0x55	0x04	0x54	0x00	none	0x05	0xAA

### 13. Read any block command

Command frame:

head	length	command	data	check	end
0x55	0x04	0x55	xx	xx	0xAA

**explanation:**

**data: xx---**Need to read the contents of the block number;

Response frame:

head	length	command	status	data	check	end
0x55	0x14	0x55	0x00	XX...XX	XX	0xAA

**explanation:**

**data: xx...xx-----**To read the content, each block of 16 byte.

Note: for example using A authentication, the partial content read out are 0X00.

#### 14. Write block command

Command frame:

head	length	command	data1	data2	check	end
0x55	0x14	0x56	xx	xx....xx	xx	0xAA

**explanation:**

data1: xx----Need to write content block;

note: not to support the key area to rewrite.

Data 2: xx...xx----the need to write the data content; each 16 bytes.

Response frame:

head	length	command	status	data	check	end
0x55	0x04	0x56	0x00	none	0x07	0xAA

Note: 9 ~ 12 functions which are sequentially passed, before it could be 13, 14, and must be in the abolition of the " automatic detection " mode of operation.

#### 15. Card type select

Command frame:

Head	Length	Command	Data 1	Bcc	End
0x55	0x04	0x42	XX	XX	0xAA

Explanation:

data 1: 0x00----4 bytes UID S50

0x01---7 bytes UID S50

0x02---7 bytes UID ultralight

Response frame:

Head	Length	Command	State	BCC	End
0x55	0x04	0x42	0x00	XX	0xAA

## 16. Get UID command

Command frame:

Head	Length	Command	Data 1	BCC	End
0x55	0x03	0x80	None	0xD6	0xAA

Response frame:

Head	Length	Command	State	Data 1	BCC	End
0x55	0x0B/0x08	0x80	0x00	XX.....X X	XX	0xAA

Explanation:

Data 1 : the 7 bytes UID or 4 bytes UID

## Attachment 1:

### BCC calculate sub-program (C language)

```
unsigned char BCC_PRO(unsigned char *databuff, unsigned char length)
{
    unsigned char BCC=0;
    unsigned char i;
    for(i=0; i<length; i++)
    {
        BCC ^= databuff[i];
    }
    return BCC;
}
```

#### a). parameter explanation

\*databuff : Pointer variable, points to the first address of the frame.

length : calculate length, the need to calculate the number of bytes, the number of bytes before the BCC for all

#### b). Function returns after the contents of the BCC, 1 byte.