LinkSprite RFID Reader/Writer Module
(ISO14443 *Proximity cards* standard)

HY502F

User Manual

December 2009

## Brief Introduction

HY502 series of RFID reader/writer modules are based on non-contact card reader ASCI chip compatible with ISO14443 standard. It uses 600nm CMOS EEPROM technology, supports ISO14443 typeA protocol, and also supports the MIFARE standard encryption algorithm. The chipset integrates analog modulation and demodulation circuits, only requires minimal peripheral circuits to function. The module supports UART interface, I2C interface, and SPI interface. The digital circuits has dual working voltages mode, TTL and CMOS. The HY502 module is targeting water, electricity, gas meters, vending machines, access control, elevators, drinking fountains, telephone billing system or other identification card reader system applications.

Users can simply select the desired interfaces to harvest the full operation of the system and do not need to struggle with the complicated radio base station design.

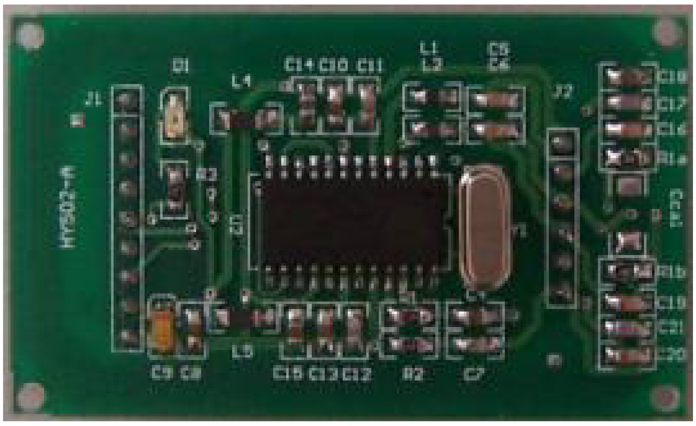
HY502 series supports Mifare One S50, S70, Ultra Light & Mifare Pro, FM11RF08 and other compatible cards. It can be set to automatically find cards, by default, to automatically find cards.

HY502 series is a low-power modules, wide-voltage 2.7 ~ 5.5V, using an integrated module with embedded antenna can significantly reduce PCB size.

## Features

- Supports three interfaces at the same time:
  - UART serial interface
  - SPI Interface
  - IIC interface
- Automatically detect the card close to the antenna area, and generate an interrupt signal to the host MCU.
- Employ chipset of ISO14434A standard, and support MIFARE standard encryption algorithm;
- Working voltage is between 2.7V- 5.5V, has TTL/CMOS voltage modes.
- Use industrial-grade high-performance processor, built-in hardware watchdog, with high reliability;
- Anti-jamming processing and excellent EMC performance;
- A few simple commands can cover the complicated underlying read and write card operations.

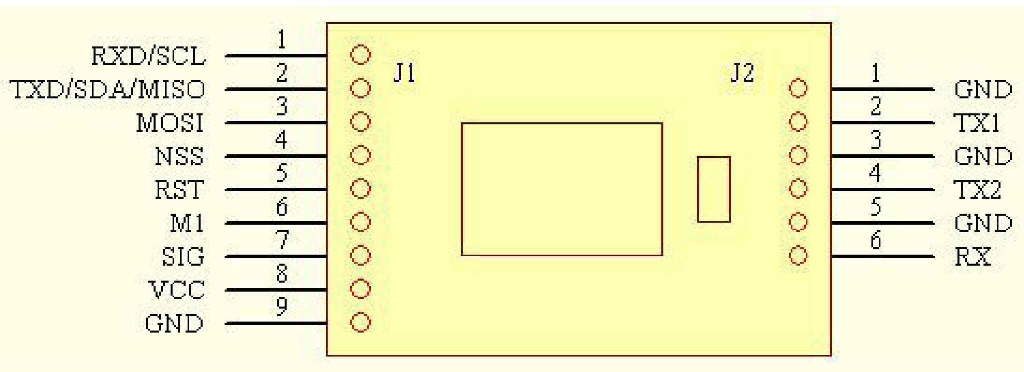
## Pictures



Size: 57.8mm X 34.5mm

The embedded antenna can read cards within 0-6cm. The external antenna, which can be connected to J2 by disconnecting the 4 short soldering point, can read cards within 0-10cm.

## Pins




## Definition of pins:

J1 is the connectors from module to host MCU, J2 is the connector for external antenna.

J1 connector:

| Pins | Name | IO type (TTL/CMOS Voltage) | Description |
|---|---|---|---|
| J1-1 | RXD/SCL | I/O | UART receive or clock line for IIC and SPI |
| J1-2 | TXD/SDA/MISO | I/O | UART send or data line for IIC, MISO for SPI |
| J1-3 | MOSI | I/O | MOSI for SPI |
| J1-4 | NSS | I | Slave selection for SPI |
| J1-5 | RST | I | Reset, and active low. Floating is okay |
| J1-6 | M1 | I | Interface selection bit 1 |
| J1-7 | SIG | O | Interrupt signal, 0 means card is present |
| J1-8 | VCC | Power | Positive terminal of power |
| J1-9 | GND | Ground | Ground |

J2 connector:

| Pin | Name | Description |
|---|---|---|
| J2-1 | GND | Ground |
| J2-2 | TX1 | Sending antenna 1 |
| J2-3 | GND | Ground |
| J2-4 | TX2 | Sending antenna 2 |
| J2-5 | GND | Ground |
| J2-6 | RX | Receive antenna |

## Electrical specification:

| Charater | Parameter | Minimum | Typical | Maximum | Unit |
|---|---|---|---|---|---|
| $T_{STR}$ | Environment or storage temperature | -40 | | +150 | $^{o}C$ |
| $T_{OP}$ | Operating temperature | -25 | +25 | +85 | $^{o}C$ |
| $V_{cc}$ | Operating voltages | 3 | 3.3* | 3.6 | V |
| | | 4.5 | 5 | 5.5 | |
| $I_{cc1}$ | Operating current | 7 | 54 | 120 | mA |
| $T_{RST}$ | Minimum reset pulse duration | 1.6 | | | us |

- In UART mode, the operating voltage is 5V. If want to work under 3.3V, please contact us at sales@cutedigi.com
- If not specified, BOD is 2.7V. BOD is 4V if specifically work at 5V.

## Selection of interface:

HY502 supports UART, IIC and SPI at the same time. The selection of interface is one by setting the voltage level of NSS and M1.

| | Interface | | |
|---|---|---|---|
| Pins | IIC | UART | SPI |
| M1 NSS | 10 | 11 | 01 |

The change of interface can be done when the module is in operation. But due to the interference, the process needs 4ms. To guarantee the reliability, we recommend to add a delay of 20ms.

## Definition of interface pins at different mode:

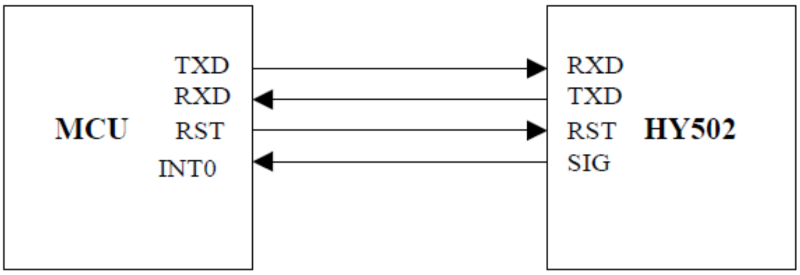
| Pin | J1-1 | J1-2 | J1-3 | J1-4 | J1-5 |
|---|---|---|---|---|---|
| Mode | RXD/SCL | TXD/SDA/MISO | MOSI | NSS | M1 |
| IIC | SCL | SDA | A0 | 0 | 1 |
| SPI | SCL | MISO | MOSI | 1 | 0 |
| UART | RXD | TXD | X | 1 | 1 |

Note: 0 means low voltage, 1 means high voltage, x means "don't care".

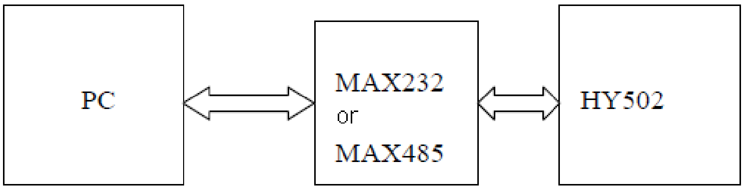
## UART interface

The default baud rate is 19200bps. Power on default is automatically scan for card, and don't need the host to frequently issue scan for card command. When card is present, SIG will send an interrupt signal, and then host MCU can use scan for card command to read the serial number of the card.

The following two diagrams show how HY502 is connected to host MCU and PC. A RS232 level shifter is needed when connecting to PC. The RST signal can be left floating, and will automatically reset on power cycle.



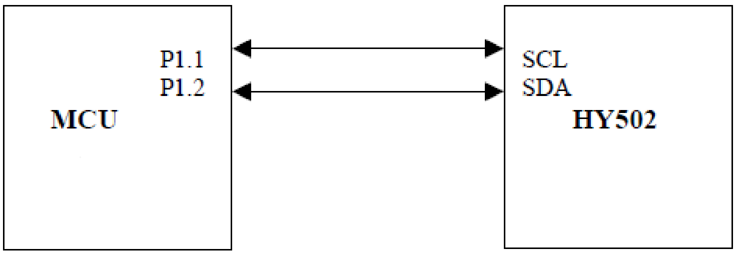

Connection between MCU and HY502.



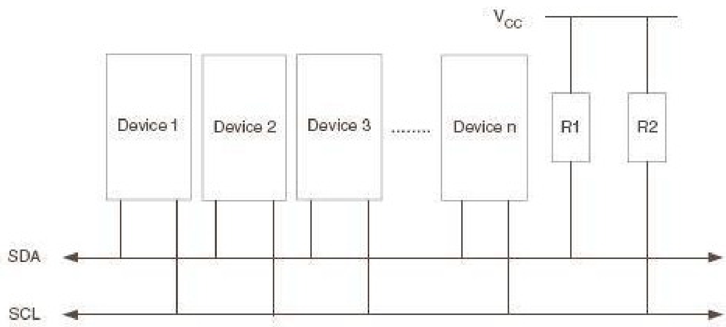

Connection between PC and HY502 using a RS232 or RS485 signal level shifter.

## IIC interface

- Only require 2 wire
- 7 bit programmable address, and can support up to 127 module
- Data rate can be as high as 400kHz.
- Noise suppression circuit can support glitch with duration shorter than 50ns.
- Internal pull-up resistors




Connection between MCU and HY502.

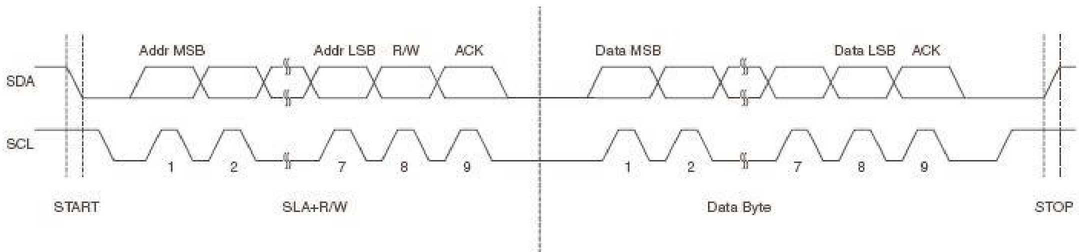



IIC multiple devices

R1 and R2 are pull-up resistors. The normal value is 10k. The driving capability is 400pF without additional driver.

IIC timing diagram:

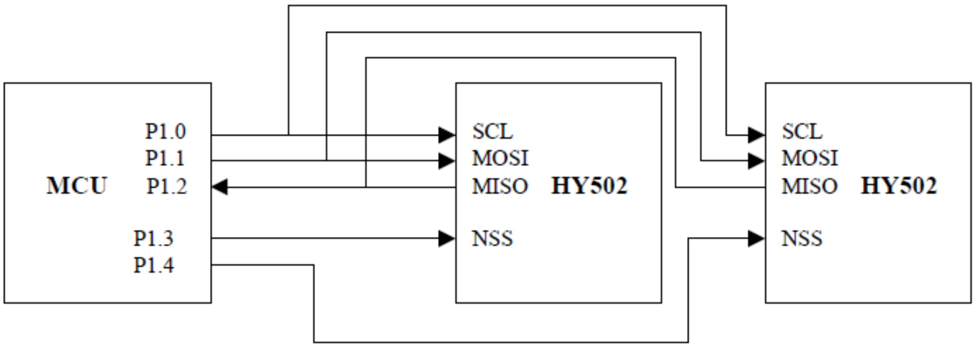
The bus timing diagram is shown below. The SDA must be kept stable logical signal when SCL is high. Logical high is data 1 and logical low is data 0. The data can only be changed when the SCL is logical low.




IIC TIME SEQUENCE

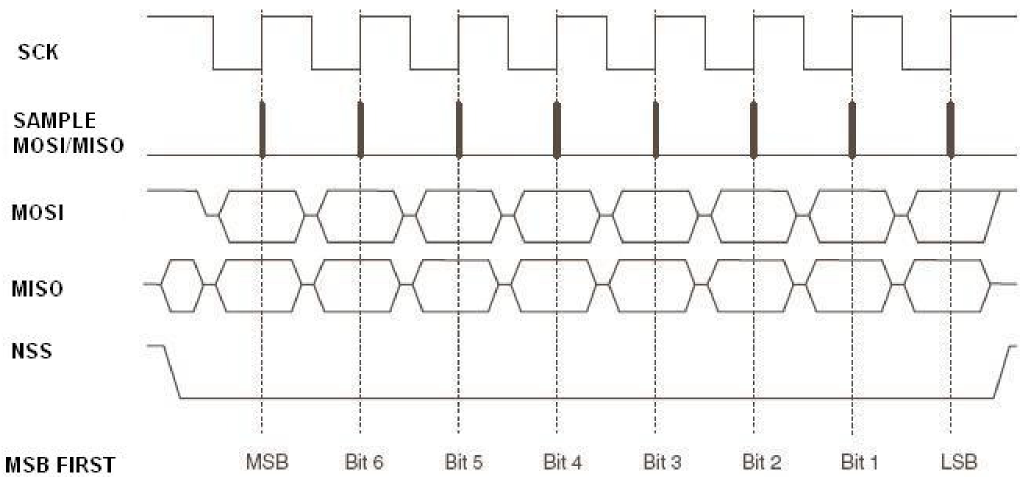
## SPI interface

- 3-wire synchronous data transfer
- Write jam protection
- Wake up call from bus
- Master and salve




NSS is the slave selection signal, and active low.

SPI timing diagram:




SPI TIME SEQUENCE

## Driver example

```
#include "main.h"
#include "hy502.h"
#define KEY_PRESS 0
sbit M1 =P1^3;
// mode define
#define UMODE 0 // uart mode
#define IMODE 1 // iic mode
#define SMODE 2 // spi mode
// mode select funtion
void modset(unsigned char xx)
```

```
{
if(xx==0) //uart
{
M1=1;
NSS=1;
}
else if(xx==1) //iic
{
M1=1;
NSS=0;
}
else if(xx==2) //SPI
{
M1=0;
NSS=1;
}
}
/************************** main function start here ******************************/
void main()
{
uchar idata cStatus;
uchar BusMode;
InitializeSystem(); // Init
Reset_HY502();
// LED_YELLOW=0; //test led
LED_GREEN=0;
delay_10ms(50);
// LED_YELLOW=1;
LED_GREEN=1; //LED light 0.5s
beep(2); //test buzz
//----- test key------//
if(KEY1==KEY_PRESS) splash(1);
if(KEY2==KEY_PRESS) splash(2);
if(KEY3==KEY_PRESS) splash(3);
if(KEY4==KEY_PRESS) splash(4);
if(KEY1==KEY_PRESS)
{
BusMode=UMODE; // Select UART
}
else if(KEY2==KEY_PRESS)
{
BusMode=IMODE ; // Select IIC
 {
a0=1;
a1=0;
a2=1;
}
}
else if(KEY3==KEY_PRESS)
{
BusMode=SMODE ; // Select SPI
 }

modset(BusMode); // Set interface mode
```

```
delay_10ms(50);
//-----------------main loop ---------------//
while (1)
{
// KeyPress(); process key
 // add your code here
//check command tag
 if (g_bReceCommandOk)
{
g_bReceCommandOk = FALSE;
if(BusMode==IMODE)
{
cStatus=cmd_process_iic();
if(cStatus)
{
cStatus=0;
}
}
else if(BusMode==SMODE)
{
cStatus=cmd_process_spi();
if(cStatus)
{
cStatus=0;
}
}
else
{
// cStatus=cmd_process_uart();
if(cStatus)
{
cStatus=0;
UartSend(g_cReceBuf); // Send data to uart.
// add your code here
}
}
```

## Interface Communication Protocols

### 1. UART protocol

- 1 start bit + 8 data bit + 1 stop bit
- Baud rate: 19200

- Command format:
    - Command header + length byte + command byte + data field + check byte
- Command header: 0XAA 0xBB, if the following data contains 0xAA, please add an additional 0x00 to distinguish the command header, but the length byte will not be increased.
- Length byte: the byte length from length byte to last byte of data field.
- Command byte: command code
- Data field: the data for the command code
- Check byte: the accumulated XOR byte value from the length byte to last byte of data field.

- Return data format:
    - Success: Command header + length byte + command byte + data field + check byte
    - Failed: :Command header + length byte + inverted command byte + data field + check byte

## 2. IIC protocol

- The 4MSB of the IIC address is 1010, ie, 0xA0, the four LSB of IIC address can be set set using A2A1A0+W/R.
- Data rate of IIC communication: 400kbps

- Data format: (Address+W/R) + length byte + command byte + data field + check byte
    - Example, the module address is 0xA0, Write command, W/R=0, so the write command will be: 0xA0 +0x0 = 0xA00
    - Example, the module address is 0xA0, Read command, W/R=1, so the read command will be: 0xA0+0x1= 0xA1
- Length byte: the length from length byte to the last byte of data field
- Command byte: the command code
- Data field: the data for the command code
- Check byte: the accumulated XOR byte value from the length byte to last byte of data field.

- Return data format:
    - Success: Command header + length byte + command byte + data field + check byte
    - Failed: Command header + length byte + inverted command byte + data field + check byte

## 3. SPI protocol

- Data format: Status byte + length byte + command byte + data field + check byte

- Status byte: the status byte of the bus. A status byte will be sent in the beginning of sending operation.
- Length byte: the length from length byte to the last byte of data field
- Command byte: the command code
- Data field: the data for the command code
- Check byte: the accumulated XOR byte value from the length byte to last byte of data field.

- Return data format:
  - Success: Command header + length byte + command byte + data field + check byte
  - Failed: Command header + length byte + inverted command byte + data field + check byte

**Command table and return value (UART is used as example, IIC and SPI will not include the command header 0xAA 0xBB)**

**Note: Unless specified, all numbers are hex.**

| ID | Command Name | Status | Length Byte | Command Code | Data and Description |
|---|---|---|---|---|---|
| 1 | Module Control | Send | 0x03 | 0x11 | 1 byte control byte, software power dropping mode, nonzero will exit soft power dropping mode, and 0x00 will enter soft power dropping mode |
| | | Return | 0x02 | 0x11 | |
| | Example | Send: aabb (Command header) 03 (length byte) 11 (command code) 00 (data field) 12 (check byte)<br>Return: aabb (Command header) 02 (length byte) 11 (command code) 13 (check byte) | | | |
| 2 | Set card IDLE | Send | 0x02 | 0x12 | When this command is executed, the card will be set to be idle. For re-activation, card needs to be removed from antenna area, and move back. |
| | | Success Return | 0x02 | 0x12 | |
| | | Fail Return | 0x02 | 0xED | |
| | Example | Send: aabb (Command header) 02 (length byte) 12 (command code) 10 (check byte)<br>Return: aabb (Command header) 02 (length byte) 12 (command code) 10 (check byte) | | | |
| 3 | Set automatic | Send | 0x03 | 0x13 | 1 byte data field, 0x01 to turn on automatic scan, and 0x00 to turn off |

| | | | | | |
|---|---|---|---|---|---|
| | scan | Success Return | 0x02 | 0x13 | |
| | | Fail Return | 0x02 | 0xEC | |
| | Example | Send: aabb 03 13 00 10<br>Command header length byte command code data field check byte<br>Return: aabb 02 13 11<br>Command header length byte command code check byte | | | |
| 4 | Read card type | Send | 0x02 | 0x19 | No data field |
| | | Success Return | 0x04 | 0x19 | 2 byte data field, card type<br>S50 card: 0x0400, S70: 0x0200. |
| | | Fail Return | 0x02 | 0xe6 | |
| | Example | Send: aabb 02 19 1b<br>Command header length byte command code check byte<br>Return: aabb 04 19 04 00 19<br>Command header length byte command code data field check byte | | | |
| 5 | Scan for card | Send | 0x02 | 0x20 | This function includes scan card, anti-collision, and select the card |
| | | Success Return | 0x06 | 0x20 | 4 byte data: the serial number of the card |
| | | Fail Return | 0x02 | 0xDF | |
| | Example | Send: aabb 02 20 22<br>Command header length byte command code check byte<br>Return: aabb 06 20 92BF7259 20<br>Command header length byte command code data field check byte | | | |
| 6 | Read block | Send | 0x0A | 0x21 | 1 byte key tag+ 1 byte block number + 6 byte key<br>Key tag: 0x00 – PICC_AYTHENT1A<br>0x01 – PICC_AUTHENT1B |
| | | Success Return | 0x12 | 0x21 | 16 byte data |
| | | Fail Return | 0x02 | 0xDE | |
| | Example | Send: aabb 0a 21 00 08 ffffffffffff 23<br>Command header length byte command code data check byte<br>Return: aabb 12 21 00112233445566778899AA00BBCCDDEEFF 33<br>Command header length byte command code data field check byte | | | |
| 7 | Write block | Send | 0x1A | 0x22 | 1 byte key tag+ 1 byte block number + 6 byte key<br>Key tag: 0x00 – PICC_AYTHENT1A<br>0x01 – PICC_AUTHENT1B |
| | | Success Return | 0x02 | 0x22 | |
| | | Fail Return | 0x02 | 0xDD | |
| | Example | Send: aabb 1a 22 00 08 ffffffffffff 00112233445566778899AA00BBCCDDEEFF 30<br>Command header length byte command code data check byte<br>Return: aabb 02 22 20<br>Command header length byte command code check byte | | | |
| 8 | Wallet Initialization | Send | 0x0E | 0x23 | 1 byte key tag+ 1 byte block number + 6 byte key + 4 bytes init wallet amount |
| | | Success | 0x02 | 0x23 | |

| | | | | | |
|---|---|---|---|---|---|
| | | Return | | | |
| | | Fail Return | 0x02 | 0xDC | |
| | Example | Send: aabb 0e 23 00 08 ffffffffffff 11110000 24<br>Command header length byte command code data check byte<br>Return: aabb 02 23 21<br>Command header length byte command code check byte | | | |
| 9 | Read wallet | Send | 0x0A | 0x24 | 1 byte key tag+ 1 byte block number + 6 byte key |
| | | Success Return | 0x06 | 0x24 | 4 bytes wallet amount (LSB in the front) |
| | | Fail Return | 0x02 | 0xDB | |
| | Example | Send: aabb 0a 24 00 09 ffffffffffff 11110000 27<br>Command header length byte command code data check byte<br>Return: aabb 06 24 11110000 22<br>Command header length byte command code data check byte | | | |
| 10 | Deposit money | Send | 0x0E | 0x25 | 1 byte key tag+ 1 byte block number + 6 byte key+ 4 bytes wallet amount (LSB in the front) |
| | | Success Return | 0x02 | 0x25 | |
| | | Fail Return | 0x02 | 0xDA | |
| | Example | Send: aabb 0e 25 00 08 ffffffffffff 11110000 22<br>Command header length byte command code data check byte<br>Return: aabb 02 25 27<br>Command header length byte command code check byte | | | |
| 11 | Withdraw money | Send | 0x0E | 0x26 | 1 byte key tag+ 1 byte block number + 6 byte key+ 4 bytes wallet amount (LSB in the front) |
| | | Success Return | 0x02 | 0x26 | |
| | | Fail Return | 0x02 | 0xD9 | |
| | Example | Send: aabb 0e 26 00 08 ffffffffffff 11110000 21<br>Command header length byte command code data check byte<br>Return: aabb 02 26 24<br>Command header length byte command code check byte | | | |

**LinkSprite Technologies, Inc.**

**Add**：1410 Cannon Mountain Dr, Longmont, CO 80503

**Tel**：  720-204-8599

**Email**：sales@linksprite.com

**Technical questions:** support@linksprite.com

**Web**：www.linksprite.com