

Unit RFID / RFID2 Arduino 使用教程

1.准备工作

- 环境配置：参考 [Arduino IDE 上手教程](#) 完成 IDE 安装，并根据实际使用的开发板安装对应的板管理与需要的驱动库。
- 使用到的驱动库：
 - [M5Unified](#)
 - [M5GFX](#)
 - [MFRC522_I2C](#)
- 使用到的硬件产品：
 - [Basic v2.7](#)
 - [Unit RFID2](#) 或 Unit RFID



2.读取UID示例

编译要求

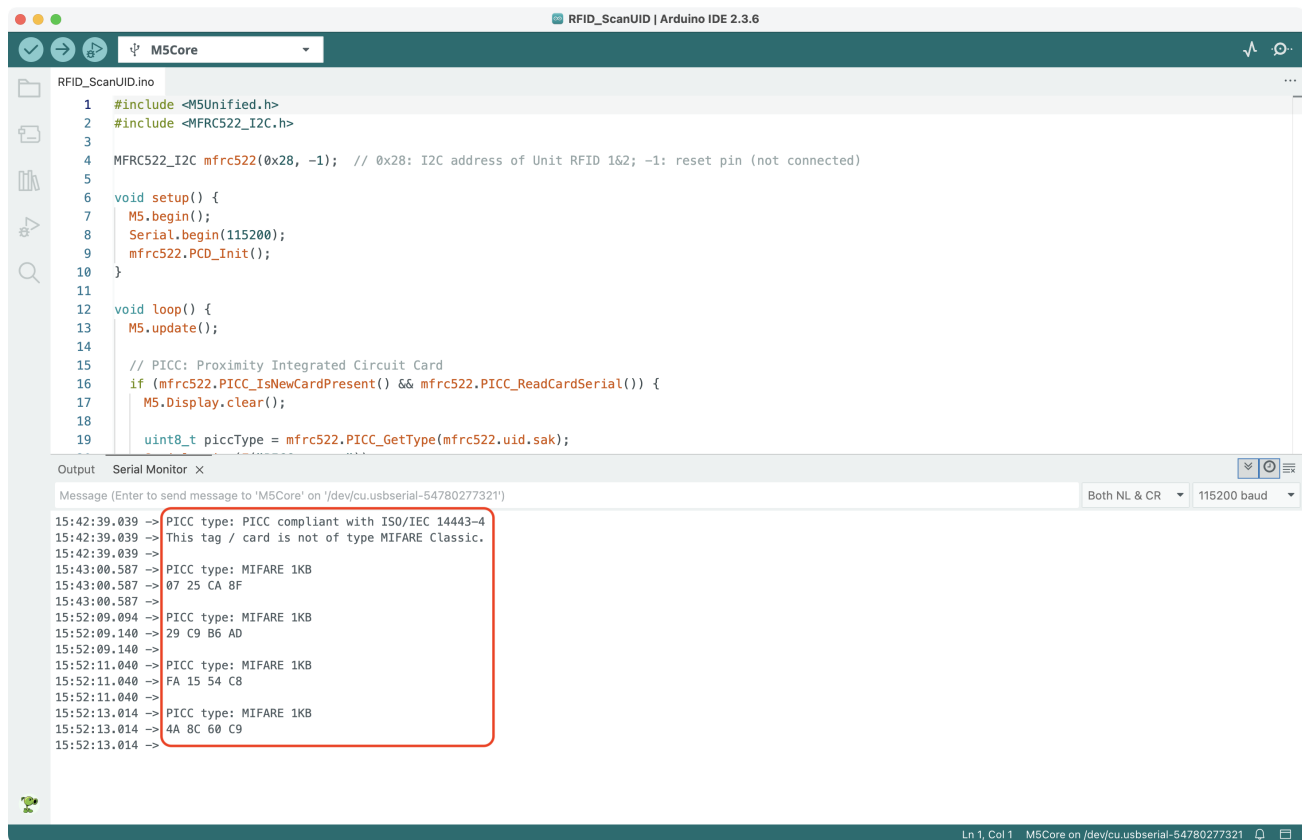
M5Stack 板管理版本 $\geq 3.2.2$

M5Unified 库版本 $\geq 0.2.8$

M5GFX 库版本 $\geq 0.2.11$

MFRC522_I2C 库版本 ≥ 1.0

```
1  #include <M5Unified.h>
2  #include <MFRC522_I2C.h>
3
4  MFRC522_I2C mfrc522(0x28, -1); // 0x28: I2C address of Unit RFID / RFID2; -1: reset pin (not connec
5
6  void setup() {
7      M5.begin();
8      Serial.begin(115200);
9      mfrc522.PCD_Init();
10 }
11
12 void loop() {
13     M5.update();
14
15     // PICC: Proximity Integrated Circuit Card
16     if (mfrc522.PICC_IsNewCardPresent() && mfrc522.PICC_ReadCardSerial()) {
17         M5.Display.clear();
18
19         uint8_t piccType = mfrc522.PICC_GetType(mfrc522.uid.sak);
20         Serial.print(F("PICC type: "));
21         Serial.println(mfrc522.PICC_GetTypeName(piccType));
22
23         // Check if the tag / card is of type MIFARE Classic
24         if (piccType != MFRC522_I2C::PICC_TYPE_MIFARE_MINI
25             && piccType != MFRC522_I2C::PICC_TYPE_MIFARE_1K
26             && piccType != MFRC522_I2C::PICC_TYPE_MIFARE_4K) {
27             Serial.println(F("This tag / card is not of type MIFARE Classic.\n"));
28             delay(500);
29             return;
30         }
31
32         // Output the stored UID data
33         for (byte i = 0; i < mfrc522.uid.size; i++) {
34             Serial.printf("%02X ", mfrc522.uid.uidByte[i]);
35         }
36         Serial.println("\n");
37         delay(500);
38     }
39 }
```



3.读写卡示例

编译要求

M5Stack 板管理版本 >= 3.2.2

M5Unified 库版本 >= 0.2.8

M5GFX 库版本 >= 0.2.11

MFRC522_I2C 库版本 >= 1.0

```
1  #include <M5Unified.h>
2  #include <MFRC522_I2C.h>
3
4  MFRC522_I2C mfrc522(0x28, -1); // 0x28: I2C address of Unit RFID / RFID2; -1: reset pin (not connected)
5  MFRC522_I2C::MIFARE_Key key;
6
7  void setup() {
8      M5.begin();
9      Serial.begin(115200);
10     mfrc522.PCD_Init();
11
12     // Prepare the key (used both as key A and as key B) with FFFFFFFFFFh,
13     // which is the default at chip delivery from the factory.
14     for (byte i = 0; i < 6; i++) {
15         key.keyByte[i] = 0xFF;
16     }
17 }
18
19 // Helper routine to dump a byte array as hex values to Serial.
20 void dump_byte_array(byte *buffer, byte bufferSize) {
21     for (byte i = 0; i < bufferSize; i++) {
22         Serial.printf("%02X ", buffer[i]);
23     }
24 }
25
26 void loop() {
27     M5.update();
28
29     // PICC: Proximity Integrated Circuit Card
30     if (mfrc522.PICC_IsNewCardPresent() && mfrc522.PICC_ReadCardSerial()) {
31         M5.Display.clear();
32
33         uint8_t piccType = mfrc522.PICC_GetType(mfrc522.uid.sak);
34         Serial.print(F("PICC type: "));
35         Serial.println(mfrc522.PICC_GetTypeName(piccType));
36
37         // Check if the tag / card is of type MIFARE Classic
38         if (piccType != MFRC522_I2C::PICC_TYPE_MIFARE_MINI
39             && piccType != MFRC522_I2C::PICC_TYPE_MIFARE_1K
40             && piccType != MFRC522_I2C::PICC_TYPE_MIFARE_4K) {
41             Serial.println(F("This tag / card is not of type MIFARE Classic.\n"));
42             delay(500);
43             return;
44         }
45
46         // Output the stored UID data
47         String uid = "";
48         for (byte i = 0; i < mfrc522.uid.size; i++) {
49             Serial.printf("%02X ", mfrc522.uid.uidByte[i]);
50             uid += String(mfrc522.uid.uidByte[i], HEX);
```



```

51 }
52 Serial.println("\n");
53
54 // mfrc522.PICC_DumpToSerial(&(mfrc522.uid));
55 // Serial.println("\n");
56
57 // In this example, we use the second sector (sector #1) including blocks #4, #5, #6, #7
58 byte sector = 1;
59 byte blockAddr = 4;
60 byte trailerBlock = 7;
61 byte dataBlock[] = {
62     0x01, 0x02, 0x03, 0x04, // 1, 2, 3, 4,
63     0x05, 0x06, 0x07, 0x08, // 5, 6, 7, 8,
64     0x09, 0x0a, 0x0b, 0x0c, // 9, 10, 11, 12,
65     0x0d, 0x0e, 0x0f, 0xff // 13, 14, 15, 255
66 };
67 MFRC522_I2C::StatusCode status;
68 byte buffer[18];
69 byte size = sizeof(buffer);
70
71 // Authenticate using key A
72 Serial.println(F("Authenticating using key A..."));
73 status = (MFRC522_I2C::StatusCode)mfrc522.PCD_Authenticate(MFRC522_I2C::PICC_CMD_MF_AUTH_KEY_A,
74 if (status != MFRC522_I2C::STATUS_OK) {
75     Serial.print(F("PCD_Authenticate() failed: "));
76     Serial.println(mfrc522.GetStatusCodeName(status));
77     return;
78 }
79
80 // Show the whole sector as it currently is
81 Serial.println(F("Current data in sector: "));
82 mfrc522.PICC_DumpMifareClassicSectorToSerial(&(mfrc522.uid), &key, sector);
83 Serial.println("");
84
85 // Read data from the block
86 Serial.print(F("Reading data from block #"));
87 Serial.print(blockAddr);
88 Serial.println(F(" ..."));
89 status = (MFRC522_I2C::StatusCode)mfrc522.MIFARE_Read(blockAddr, buffer, &size);
90 if (status != MFRC522_I2C::STATUS_OK) {
91     Serial.print(F("MIFARE_Read() failed: "));
92     Serial.println(mfrc522.GetStatusCodeName(status));
93 }
94 Serial.print(F("Data in block #"));
95 Serial.print(blockAddr);
96 Serial.println(F(": "));
97 dump_byte_array(buffer, 16);
98 Serial.println("\n");
99
100 // Authenticate using key B
101 Serial.println(F("Authenticating again using key B..."));
102 status = (MFRC522_I2C::StatusCode)mfrc522.PCD_Authenticate(MFRC522_I2C::PICC_CMD_MF_AUTH_KEY_B,
103 if (status != MFRC522_I2C::STATUS_OK) {

```

```

104     Serial.print(F("PCD_Authenticate() failed: "));
105     Serial.println(mfrc522.GetStatusCodeName(status));
106     return;
107 }
108
109 // Write data to the block
110 Serial.print(F("Writing data into block #"));
111 Serial.print(blockAddr);
112 Serial.println(F(" ..."));
113 dump_byte_array(dataBlock, 16);
114 Serial.println();
115 status = (MFRC522_I2C::StatusCode)mfrc522.MIFARE_Write(blockAddr, dataBlock, 16);
116 if (status != MFRC522_I2C::STATUS_OK) {
117     Serial.print(F("MIFARE_Write() failed: "));
118     Serial.println(mfrc522.GetStatusCodeName(status));
119 }
120 Serial.println("");
121
122 // Read data from the block again, now it should be what we have written
123 Serial.print(F("Reading data from block #"));
124 Serial.print(blockAddr);
125 Serial.println(F(" ..."));
126 status = (MFRC522_I2C::StatusCode)mfrc522.MIFARE_Read(blockAddr, buffer, &size);
127 if (status != MFRC522_I2C::STATUS_OK) {
128     Serial.print(F("MIFARE_Read() failed: "));
129     Serial.println(mfrc522.GetStatusCodeName(status));
130 }
131 Serial.print(F("Data in block #"));
132 Serial.print(blockAddr);
133 Serial.println(F(": "));
134 dump_byte_array(buffer, 16);
135 Serial.println("\n");
136
137 // Check if the data in block is what we have written, by counting the number of bytes that are
138 Serial.println(F("Checking result..."));
139 byte count = 0;
140 for (byte i = 0; i < 16; i++) {
141     // Compare buffer (what we've read) with dataBlock (what we've written)
142     if (buffer[i] == dataBlock[i]) {
143         count++;
144     }
145 }
146 Serial.print(F("Number of bytes that match = "));
147 Serial.println(count);
148 if (count == 16) {
149     Serial.println(F("Success :-"));
150 } else {
151     Serial.println(F("Failure, no match :-("));
152     Serial.println(F(" perhaps the write didn't work properly..."));
153 }
154 Serial.println();
155
156 // Dump the sector data

```

```

157 Serial.println(F("Current data in sector: "));
158 mfrc522.PICC_DumpMifareClassicSectorToSerial(&(mfrc522.uid), &key, sector);
159 Serial.println("");
160
161 // Halt PICC
162 mfrc522.PICC_HaltA();
163 // Stop encryption on PCD
164 mfrc522.PCD_StopCrypto1();
165
166 Serial.println("=====");
167 }
168 }

```

这段程序会在完成 Key A 认证后读取 Sector 1 中的 Block 4 的数据，然后在完成 Key B 认证后修改 Sector 1 中的 Block 4 的数据。在这个过程中标签 / 卡片需要保持靠近 Unit RFID2。程序输出如下：

```

RFID_ReadWrite.ino
1 #include <M5Unified.h>
2 #include <MFRC522_I2C.h>
3
4 MFRC522_I2C mfrc522(0x28, -1); // 0x28: I2C address of Unit RFID / RFID2; -1: reset pin (not connected)
5 MFRC522_I2C::MIFARE_Key key;
6
7 void setup() {
  Output Serial Monitor x
  Message (Enter to send message to 'M5Core' on '/dev/cu.usbserial-54780277321') Both NL & CR 115200 baud
  17:11:17.881 -> PICC type: MIFARE 1KB
  17:11:17.913 -> 4A 8C 60 C9
  17:11:17.913 ->
  17:11:17.913 -> Authenticating using key A...
  17:11:17.913 -> Current data in sector:
  17:11:17.913 -> 1 7 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF [ 0 0 1 ]
  17:11:17.946 -> 6 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [ 0 0 0 ]
  17:11:17.946 -> 5 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [ 0 0 0 ]
  17:11:17.977 -> 4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [ 0 0 0 ]
  17:11:18.010 ->
  17:11:18.010 -> Reading data from block #4 ...
  17:11:18.010 -> Data in block #4:
  17:11:18.010 -> 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 before
  17:11:18.010 ->
  17:11:18.010 -> Authenticating again using key B...
  17:11:18.010 -> Writing data into block #4 ...
  17:11:18.043 -> 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F FF
  17:11:18.043 ->
  17:11:18.043 -> Reading data from block #4 ...
  17:11:18.075 -> Data in block #4:
  17:11:18.075 -> 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F FF after
  17:11:18.075 ->
  17:11:18.075 -> Checking result...
  17:11:18.075 -> Number of bytes that match = 16
  17:11:18.075 -> Success :-))
  17:11:18.075 ->
  17:11:18.075 -> Current data in sector:
  17:11:18.075 -> 1 7 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF [ 0 0 1 ]
  17:11:18.108 -> 6 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [ 0 0 0 ]
  17:11:18.140 -> 5 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [ 0 0 0 ]
  17:11:18.140 -> 4 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F FF [ 0 0 0 ]
  17:11:18.173 ->
  17:11:19.262 -> =====
  
```

4.Init API

以下为常用 API 的使用说明。一般读写 MIFARE 卡的流程为：

1. 初始化 RFID
2. 检测新卡存在，获取卡 UID
3. 通过 UID 选中卡片，进入 active 状态
4. 通过 Key A、Key B 解锁对应的 Block
5. 读 / 写数据
6. 控制卡进入休眠状态

操作返回值状态码

```
1  enum StatusCode {
2      STATUS_OK           = 1,  // Success
3      STATUS_ERROR        = 2,  // Error in communication
4      STATUS_COLLISION    = 3,  // Collision detected
5      STATUS_TIMEOUT      = 4,  // Timeout in communication
6      STATUS_NO_ROOM      = 5,  // The buffer is not big enough
7      STATUS_INTERNAL_ERROR = 6,  // Internal error in the code. Should not happen ;- )
8      STATUS_INVALID       = 7,  // Invalid argument
9      STATUS_CRC_WRONG     = 8,  // The CRC_A does not match
10     STATUS_MIFARE_NACK    = 9   // The MIFARE PICC responded with NAK
11 };
```

实例化

函数原型：

```
MFRC522_I2C(byte chipAddress, byte resetPowerDownPin);
```

功能说明：

- 创建一个 RFID 实例

传入参数：

- byte chipAddress
 - 模块的 I2C 地址 (0x28)
- byte resetPowerDownPin
 - 主控连接到模块的复位引脚，由于未连接所以设置为 -1

返回值：

- MFRC522_I2C 类的对象

PCD_Init

函数原型：

```
void PCD_Init();
```

功能说明：

- 初始化读写器

传入参数：

- null

返回值：

- null

PICC_IsNewCardPresent

函数原型:

```
bool PICC_IsNewCardPresent();
```

功能说明:

- 扫描是否存在未检测过且处于IDLE状态的卡片，处于HALT状态的卡片将被忽略。

传入参数:

- null

返回值:

- bool
 - true: 扫描到了新卡
 - false: 未扫描到新卡

PICC_ReadCardSerial

函数原型:

```
bool PICC_ReadCardSerial();
```

功能说明:

- 读取卡 UID，读取成功后的 UID 可以在类成员Uid uid;中读取。读取操作前需执行PICC_IsNewCardPresent()或PICC_RequestA()或PICC_WakeupA()确保读取到卡片。

cpp

```
1 for (byte i = 0; i < mfrc522.uid.size; i++) {
2     Serial.printf("%02X ", mfrc522.uid.uidByte[i]);
3 }
```

传入参数:

- null

返回值:

- bool
 - true: 读取成功
 - false: 读取失败

PICC_RequestA

函数原型:

```
uint8_t PICC_RequestA(uint8_t *bufferATQA, uint8_t *bufferSize);
```

功能说明:

- 扫描检测读取范围内的 Type A 标准的卡片

传入参数:

- uint8_t *bufferATQA
 - 存储请求响应 ATQA (Answer to request) 的 buffer
- uint8_t *bufferSize
 - buffer 长度 (>2 byte)

返回值:

- uint8_t
 - StatusCode

| PICC_WakeupA

函数原型:

```
uint8_t PICC_WakeupA(uint8_t *bufferATQA, uint8_t *bufferSize);
```

功能说明:

- 唤醒范围内的 Type A 标准的卡片

传入参数:

- uint8_t *bufferATQA
 - 存储请求响应 ATQA (Answer to request) 的 buffer
- uint8_t *bufferSize
 - buffer 长度 (>2 byte)

返回值:

- uint8_t
 - StatusCode

| PICC_Select

函数原型:

```
uint8_t PICC_Select(Uid *uid, uint8_t validBits = 0);
```

功能说明:

- 通过 uid 选中一张卡片进入 active 状态

传入参数:

- Uid *uid
 - 通过扫描获取到的卡片 uid 结构体指针
- uint8_t validBits
 - 最后一个 uint8_t 中的有效位, 0 表示 8 个有效位

返回值:

- uint8_t
 - StatusCode

| PICC_HaltA

函数原型:

```
uint8_t PICC_HaltA();
```

功能说明:

- 选中当前卡片进入休眠状态

传入参数:

- null

返回值:

- uint8_t
 - StatusCode

5.MIFARE API

PCD_Authenticate

函数原型:

```
uint8_t PCD_Authenticate(uint8_t command, uint8_t blockAddr, MIFARE_Key *key, Uid *uid);
```

功能说明:

- 进行 MIFARE 身份验证。在调用此函数之前，需要对卡片进行 select 操作使其处于 active 状态。经过身份验证的 PICC 通信结束后需调用 PCD_StopCrypto1()，否则无法开始新的通信。

传入参数:

- uint8_t command
 - PICC_CMD_MF_AUTH_KEY_A
 - PICC_CMD_MF_AUTH_KEY_B
- uint8_t blockAddr
 - Block 地址
- MIFARE_Key *key
 - 默认状态下，Key A、Key B 均为FFFFFFFFFFFF
- Uid *uid

返回值:

- uint8_t
 - StatusCode

PCD_StopCrypto1

函数原型:

```
void PCD_StopCrypto1();
```

功能说明:

- 退出 PCD 的认证状态。经过身份验证的 PICC 通信结束后需调用PCD_StopCrypto1()，否则无法开始新的通信。

传入参数:

- null

返回值:

- null

| MIFARE_Read

函数原型:

```
uint8_t MIFARE_Read(uint8_t blockAddr, uint8_t *buffer, uint8_t *bufferSize);
```

功能说明:

- 从指定 blockAddr 读取数据

传入参数:

- uint8_t blockAddr
 - 实际卡片扇区中的 block 地址
- uint8_t *buffer
 - 接收数据的 buffer 指针
- uint8_t *bufferSize
 - 接收数据的 buffer 长度 (>=18 byte)

返回值:

- uint8_t
 - StatusCode

| MIFARE_Write

函数原型:

```
uint8_t MIFARE_Write(uint8_t blockAddr, uint8_t *buffer, uint8_t bufferSize);
```

功能说明:

- 向指定 blockAddr 写入数据

传入参数:

- uint8_t blockAddr
 - 实际卡片扇区中的 block 地址
- uint8_t *buffer
 - 写入数据的 buffer 指针
- uint8_t *bufferSize
 - 写入数据的 buffer 长度 (16 byte)

返回值:

- uint8_t
 - StatusCode

6. 参考链接

- 更多相关的 API 可以参考 [MFRC522_I2C 库](#)
- [ISO/IEC 14443 - Wikipedia](#)
- [MFRC522 - NXP Docs](#) (Standard performance MIFARE and NTAG frontend)
- [MFRC523 - NXP Docs](#) (Standard performance ISO/IEC 14443 A/B frontend)
- [MIFARE Classic EV1 1K - NXP Docs](#) ([中文版](#))
- [MIFARE Classic EV1 4K - NXP Docs](#)
- [AN1305 - NXP Docs](#) (MIFARE Classic as NFC Type MIFARE Classic Tag)
- [AN10833 - NXP Docs](#) (MIFARE type identification procedure)
- [AN10834 - NXP Docs](#) (MIFARE ISO/IEC 14443 PICC selection)