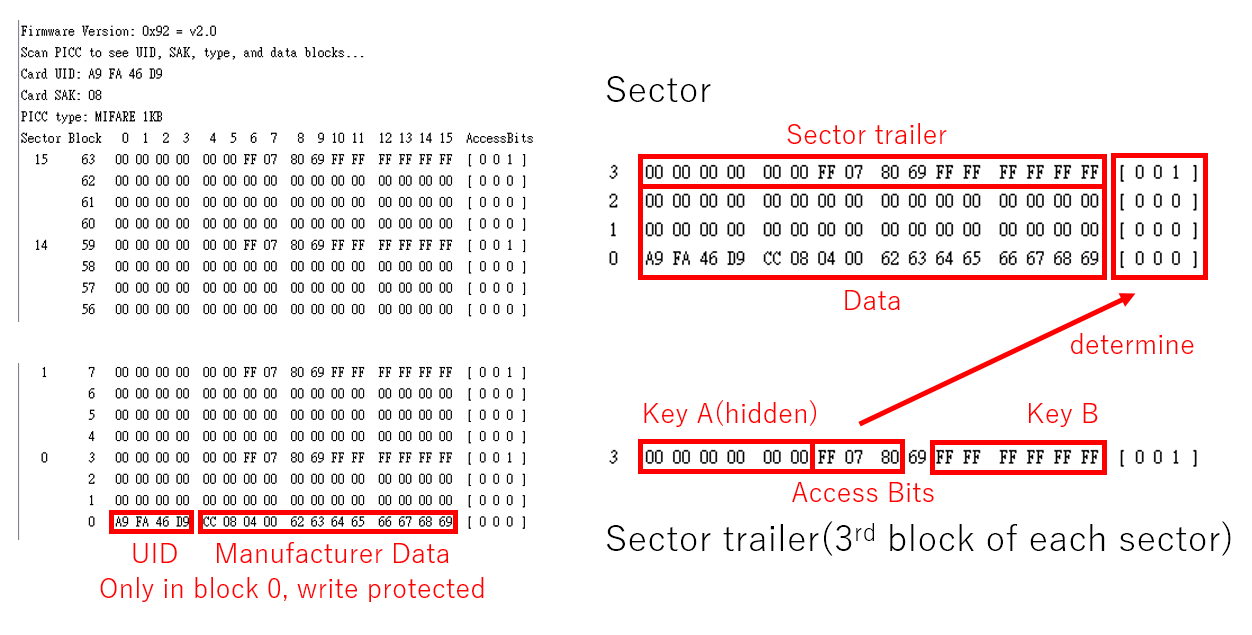
A. MIFARE Classic 1kB的資料格式

內建函數：DumpInfo

* Arduino IDE工具列點選=>檔案=>範例=>MFRC522=>DumpInfo
* 將程式上傳開發版並打開監控視窗
* 感應卡片即可看到卡片中儲存的所有資料(若通過驗證)
* 注意須等資料全部印出後再移開卡片
* 若成功讀取將看到如左下圖的輸出



畫面中可以看到關於這張卡片的資料  
其中的UID就是每張卡片的序號，就像身分證一樣，每張卡片出廠就有一個UID，理論上都是不一樣的

再來可以看到PICC type確實是MIFARE 1KB

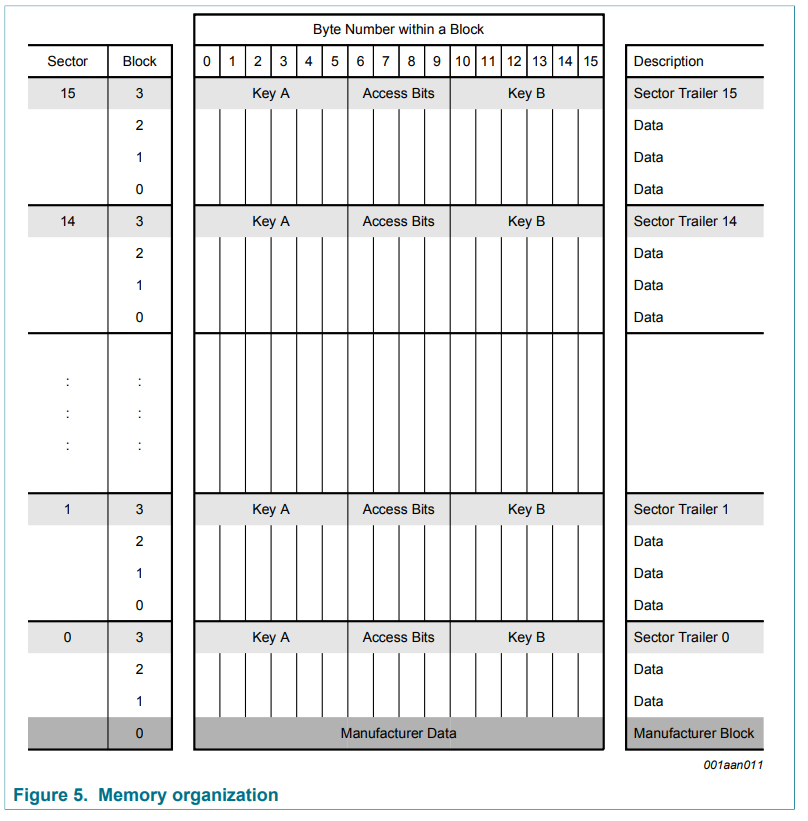
接下來就是卡片裡儲存的全部資料  
我們可以看到最左邊顯示了資料的sector，總共分成16個sector，而每個sector再分成4個block，所以總共是64個block，每個block 16bytes的資料，64個block總共就是1024bytes也就是1KB，而最右邊的accessbits則是代表這個block的access權限

接下來看一下最下面第0個sector的第0個block，前4個byte紀錄了這張卡的UID，剩下的12個byte則是生產資料，一般來說這個block是被保護住無法寫入資料做更改的

接下來看一下每個sector的結構(如右上半圖)，其中前3個block為儲存資料的data block，而最後一個block則稱作sector trailer，用來管理這個sector的access權限   
每個sector有兩把6bytes的key，其中keyA存在sector trailer的前六位，而keyB是存在最後6位，注意這邊keyA預設是看不到的，理論上來說出廠預設的keyA跟keyB都是六位FF，但這邊keyA看不到只會顯示全0，而keyB則正常顯示出全FF

sector trailer中間的3個byte是用來控制這個sector的access權限，也就是決定了剛剛看到的access bits  
剩下一個byte目前沒有使用

下圖為NXP提供的datasheet中說明資料儲存的示意圖



B. 讀取特定block的資料

#include <**SPI**.h> //引入SPI相關header

#include <**MFRC522**.h> //引入RC522相關header

#define RST\_PIN 9 //設定RST腳位

#define SS\_PIN 10 //設定SS腳位

**MFRC522** \*mfrc522; //宣告MFRC522指標，我們將透過這個物件指標對RC522進行操作

**MFRC522**::**MIFARE\_Key** \*key; //key指向的位置用來儲存key

byte bufferSize; //buffer[]的大小

byte blockNum; //要讀取的block的編號

byte \*buffer; //讀出的資料會儲存在buffer[]裡

**MFRC522**::**StatusCode** status; //暫存函數回傳的狀態值

void setup() {

**Serial**.begin(9600); //設定鮑率

**SPI**.begin(); //開始SPI介面\*

mfrc522 = new **MFRC522**(SS\_PIN, RST\_PIN); //將SS, RST傳入constructer

mfrc522->PCD\_Init() //初始化SPI介面

key = new **MFRC522**::MIFARE\_Key(); //建立key的空間

for(byte i = 0; i < 6; ++i) {

key->keyByte[i] = 0xFF; //將key設定成出廠預設全FF

}

bufferSize = 18; //設定buffersize

blockNum = 28; //這次以讀取block28作為示範

**Serial**.println(F("Read block data on a MIFARE PICC:"));

}

void loop() {

if ( ! mfrc522->PICC\_IsNewCardPresent()) //檢查是否感應到新卡片

goto FuncEnd;

if ( ! mfrc522->PICC\_ReadCardSerial()) //檢查是否成功讀取

goto FuncEnd;

**Serial**.println(F("\*\*Card Detected:\*\*"));

status = **MFRC522**::STATUS\_OK; //初始化status

status = (**MFRC522**::**StatusCode**) mfrc522->PCD\_Authenticate(

MFRC522::PICC\_CMD\_MF\_AUTH\_KEY\_A,

blockNum,

key,

&(mfrc522->uid)

); //驗證讀取sector的key

if (status != **MFRC522**::STATUS\_OK) {

**Serial**.println(F("Authentication failed:"));

**Serial**.println(mfrc522->GetStatusCodeName(status));

goto FuncEnd;

} //若失敗則印出錯誤訊息

else **Serial**.println(F("Authentication success.")); //驗證成功回傳STATUS\_OK

buffer = new byte[bufferSize]; //建立buffer空間

status = (**MFRC522**::**StatusCode**) mfrc522->MIFARE\_Read(

blockNum,

buffer,

&bufferSize

); //從block讀取資料

if (status != **MFRC522**::STATUS\_OK) {

**Serial**.print(F("MIFARE\_Read() failed: "));

**Serial**.println(mfrc522->GetStatusCodeName(status));

goto FuncEnd;

} //若失敗則印出錯誤訊息

else **Serial**.println(F("Data read success.")); //讀取成功回傳STATUS\_OK

for(byte i = 0; i < bufferSize; ++i) {

**Serial**.print(buffer[i] < 0x10 ? "0" : "");

**Serial**.print(buffer[i], HEX);

**Serial**.print(' ');

}

**Serial**.print("\n\n"); //印出讀到的資料

mfrc522->PICC\_HaltA();

mfrc522->PCD\_StopCrypto1();

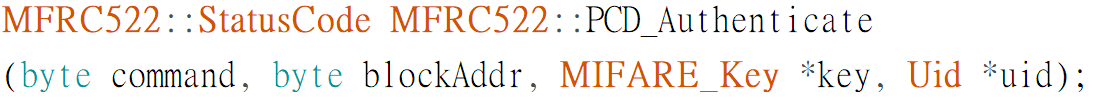
delete[] buffer;

buffer = NULL;

FuncEnd:;

}

幾個重要的函數：



|  |  |  |  |
| --- | --- | --- | --- |
| **PICC\_Command** | **Value** | **PICC\_Command** | **value** |
| PICC\_CMD\_REQA | 0x26 | PICC\_CMD\_MF\_AUTH\_KEY\_B | 0x61 |
| PICC\_CMD\_WUPA | 0x52 | PICC\_CMD\_MF\_READ | 0x30 |
| PICC\_CMD\_CT | 0x88 | PICC\_CMD\_MF\_WRITE | 0xA0 |
| PICC\_CMD\_SEL\_CL1 | 0x93 | PICC\_CMD\_MF\_DECREMENT | 0xC0 |
| PICC\_CMD\_SEL\_CL2 | 0x95 | PICC\_CMD\_MF\_INCREMENT | 0xC1 |
| PICC\_CMD\_SEL\_CL3 | 0x97 | PICC\_CMD\_MF\_RESTORE | 0xC2 |
| PICC\_CMD\_HLTA | 0x50 | PICC\_CMD\_MF\_TRANSFER | 0xB0 |
| PICC\_CMD\_RATS | 0xE0 | PICC\_CMD\_UL\_WRITE | 0xA2 |
| PICC\_CMD\_MF\_AUTH\_KEY\_A | 0x60 |  |  |

[說明請見：](https://github.com/miguelbalboa/rfid/blob/1.3.6/src/MFRC522.h)[https://github.com/miguelbalboa/rfid/blob/1.4.3/src/MFRC522.h#L255](https://github.com/miguelbalboa/rfid/blob/1.4.3/src/MFRC522.h)

首先要注意的是傳入的型別，數值跟指標不要搞錯

第一個command要傳入的是這一次操作的指令，例如這次輸入PICC\_CMD\_MF\_AUTH\_KEY\_A的意思就是說用keyA來認證，而這些指令其實就是一個數字(如上表)，例如我們這次輸入的PICC\_CMD\_MF\_AUTH\_KEY\_A，函數看到這一串之後會自動把它變成0x60  
blockAddr是要驗證的block號碼，我們剛剛有看到每個block可以單獨受到sector trailer控制讀取權限，資料的讀寫也是以一個block為單位，所以我們要做的不是一次驗證一整張卡片而是一個block

Key就是用來驗證的key，記得要傳入的是指標

Uid則是要驗證的那張卡的UID，要傳入UID的原因是如果把多張卡疊在一起一起感應，機器要如何知道這次要驗證的是哪張卡

所以說，MIFARE 1KB其實是有多張同時讀寫的能力的



注意這裡的bufferSize需傳入byte pointer

blockAddr是要讀取的block編號  
因為MIFARE\_Read一次只能讀出16bytes的資料，而傳出的資料為16data bytes+2CRC bytes，所以buffersize不能小於18

無論是讀還是寫都是以1個block為單位，一次讀出一整個block，一次也是寫入一整個block  
操作時須小心segmentation fault

注意bufferSize要傳入指標

C. 將資料寫入block



將buffer[]裡的資料寫進指定的block

基本上跟read一樣，只是bufferSize要傳入的是數值而不是指標

注意：

一次只能固定寫入16bytes

buffer的長度最小需要16bytes

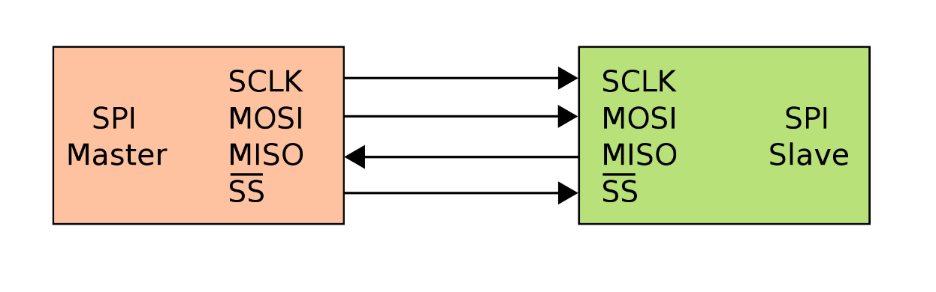
若buffer的長度大於16bytes依舊只會寫入16bytes

使用範例見示範code

* Serial Peripheral Interface
* RC522使用SPI介面與arduino連接
* 接腳有固定限制，如左圖所示，  
  所以不能任意更改，須按照接線表連接
* 更多關於SPI：  
  <https://en.wikipedia.org/wiki/Serial_Peripheral_Interface>

<https://commons.wikimedia.org/wiki/File:SPI_single_slave.svg>

<https://makerpro.cc/wp-content/uploads/2016/07/2016071101.jpg>

補充：SPI介面

* Serial Peripheral Interface
* RC522使用SPI介面與arduino連接
* 接腳有固定限制，如左圖所示，  
  所以不能任意更改，須按照接線表連接
* 更多關於SPI：  
  <https://en.wikipedia.org/wiki/Serial_Peripheral_Interface>

