



RFID

MANIPULATION COURS SMRT CARD

Manuel Technique

Module : SÉCURITÉ PHYSIQUE ET CARTES À PUCE

Réalisée par : Benabbi Saâd

Touahri Manal

Lamachi Ayman

Wahib Noura

Encadrée par : Mr. SENHADJI

Première Application

La première manipulation consiste à utiliser la Smart Card comme **un portemonnaie**.

L'application est accessible avec un **login & mot de passe**. Après authentification une interface affiche 3 champs : **solde, crédit, débit**. Notre application permet :

- L'affichage du solde qui est dans la carte.
- Créditer le solde de la carte par une somme à saisir dans le champ crédit.
- Débitier le solde de la carte par une somme à saisir dans le champ débit.
- Vérifier le nouveau solde.

1-1-Sur Arduino :

La première chose est **d'initialiser la communication** et écrire key qu'on va utiliser dans l'authentification FF FF FF FF FF , c'est la clé par défaut :

```
#include <SPI.h>
#include <MFRC522.h>
#define RST_PIN      9           // Configurable, see typical pin layout above
#define SS_PIN       10          // Configurable, see typical pin layout above
MFRC522 mfrc522(SS_PIN, RST_PIN); // Create MFRC522 instance.

MFRC522::MIFARE_Key key;
bool onetime;
void setup() {
  Serial.begin(9600); // Initialize serial communications with the PC
  while (!Serial);    // Do nothing if no serial port is opened (added for Arduinos based on ATMEGA32U4)
  SPI.begin();        // Init SPI bus
  mfrc522.PCD_Init(); // Init MFRC522 card
  onetime = true;
  // Prepare the key (used both as key A)
  // using FFFFFFFFh which is the default at chip delivery from the factory
  for (byte i = 0; i < 6; i++) {
    key.keyByte[i] = 0xFF;
  }

  Serial.println(F("Scan a MIFARE Classic PICC to demonstrate Value Block mode."));
  Serial.print(F("Using key (for A):"));

  dump_byte_array(key.keyByte, MFRC522::MF_KEY_SIZE);
  Serial.println();

  Serial.println(F("BEWARE: Data will be written to the PICC, in sector #1"));
}
```

Après il va attendre la présence d'une carte et vérifier la compatibilité de cette carte :

```
// Reset the loop if no new card present on the sensor/reader. This saves the entire process when idle.
if ( ! mfrc522.PICC_IsNewCardPresent())
  return;

// Select one of the cards
if ( ! mfrc522.PICC_ReadCardSerial())
  return;

// Show some details of the PICC (that is: the tag/card)
Serial.print(F("Card UID:"));

dump_byte_array(mfrc522.uid.uidByte, mfrc522.uid.size);
Serial.println();
Serial.print(F("PICC type: "));
MFRC522::PICC_Type piccType = mfrc522.PICC_GetType(mfrc522.uid.sak);
String text2 = mfrc522.PICC_GetTypeName(piccType);
Serial.println(mfrc522.PICC_GetTypeName(piccType));
// Check for compatibility
if (
  piccType != MFRC522::PICC_TYPE_MIFARE_MINI
  && piccType != MFRC522::PICC_TYPE_MIFARE_1K
  && piccType != MFRC522::PICC_TYPE_MIFARE_4K) {
  Serial.println(F("This sample only works with MIFARE Classic cards."));
  return;
}
```

Après il va lancer ce code une seule fois : **onetime** (boolean), et il va s'authentifier au block 4 utilisant **PCD Authenticate**, c'est le block qui contient le solde.

Il va utiliser **MIFARE GetValue** :

```
/* ===== */
if(ontime){
    byte sector = 1;
    byte valueBlockA = 4;
    byte trailerBlock = 7;
    MFRC522::StatusCode status;
    byte buffer[18];
    byte size = sizeof(buffer);
    int32_t value;

    // Authenticate using key A
    Serial.println(F("Authenticating using key A..."));
    status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, trailerBlock, &key, &(mfrc522.uid));
    if (status != MFRC522::STATUS_OK) {
        Serial.print(F("PCD Authenticate() failed: "));
        Serial.println(mfrc522.GetStatusCodeName(status));
        return;
    }

    status = mfrc522.MIFARE_GetValue(valueBlockA, &value);
    if (status != MFRC522::STATUS_OK) {
        Serial.print(F("mifare_GetValue() failed: "));
        Serial.println(mfrc522.GetStatusCodeName(status));
        return;
    }

    Serial.print("solde");
    Serial.print(valueBlockA);
    Serial.print("=");
    Serial.print(value);
    Serial.println("=tx19");

    mfrc522.PCD_StopCrypto1();
    ontime=false;
}
```

block 4

Après il va attendre un text en utilisant **Serial.readString()** et il va utiliser **split** pour avoir la commande qu'il va executer :

```
String text = Serial.readString();

int n = text.length();
char char_array[n + 1];
strcpy(char_array, text.c_str());
Serial.write(char_array);
int i = 0;
char *p = strtok (char array, ",");
char *array[3];

while (p != NULL)
{
    array[i++] = p;
    p = strtok (NULL, ",");
}
// Serial.println(array[0]);
String str=array[0];
char str1[20];
strcpy(str1, str.c_str());
// char str1[20]=str;
char str2[20]="INC";
char str3[20]="DEC";
char str4[20]="LOG";
char str5[20]="CHN";
```

pour lire le text depuis l'app

la valeur de str1 depuis l'application

pour incrementer

pour decrementer

pour Login

pour changer mdp

Après dans cette comparaison s'il trouve que **str1** et **str2** sont **identique** il va **incrémenter** en premier. Il va obtenir la valeur dU **secteur 1** et **block 4** c'est la valeur de solde et en utilisant **Mefari_GetValue** :

```
if( strcmp(str1,str2)==0) { // if str2 = INC
    int incr = atoi( array[1]);
    Serial.print(incr);
    // In this sample we use the second sector,
    // that is: sector #1, covering block #4 up to and including block #7
    byte sector = 1;
    byte valueBlockA = 4;
    // byte valueBlockB = 6;
    byte trailerBlock = 7;
    MFRC522::StatusCode status;
    byte buffer[18];
    byte size = sizeof(buffer);
    int32_t value;

    // Authenticate using key A
    Serial.println(F("Authenticating using key A..."));
    status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, trailerBlock, &key, &(mfrc522.uid));
    if (status != MFRC522::STATUS_OK) {
        Serial.print(F("PCD_Authenticate() failed: "));
        Serial.println(mfrc522.GetStatusCodeName(status));
        return;
    }

    status = mfrc522.MIFARE_GetValue(valueBlockA, &value);
    if (status != MFRC522::STATUS_OK) {
        Serial.print(F("mifare_GetValue() failed: "));
        Serial.println(mfrc522.GetStatusCodeName(status));
        return;
    }

    Serial.print("solde");
    Serial.print(valueBlockA);
    Serial.print("=");
    Serial.print(value);
    Serial.println("=tx19");
```

Annotations dans l'image :

- Une flèche rouge pointe vers la ligne `byte valueBlockA = 4;`.
- Une flèche rouge pointe vers la ligne `Serial.print(value);` avec le texte "valeur de solde" à côté.

Ensuite il va utiliser **increment** puisque dans ce cas **str1=INC** pour incrémenter et il va utiliser **MIFARE_Transfer** puisque MIFARE_Increment stocke seulement la valeur dans la mémoire volatile :

```
status = mfrc522.MIFARE_Increment(valueBlockA, incr);
if (status != MFRC522::STATUS_OK) {
    Serial.print(F("MIFARE_Increment() failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
}

status = mfrc522.MIFARE_Transfer(valueBlockA);
if (status != MFRC522::STATUS_OK) {
    Serial.print(F("MIFARE_Transfer() failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
}

// Show the new value of valueBlockA
status = mfrc522.MIFARE_GetValue(valueBlockA, &value);
if (status != MFRC522::STATUS_OK) {
    Serial.print(F("mifare_GetValue() failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
}

Serial.print("solde");
Serial.print(valueBlockA);
Serial.print("=");
Serial.print(value);
Serial.println("=tx19");

mfrc522.PCD_StopCrypto1();
}else{
    mfrc522.PCD_StopCrypto1();
}
```

Après dans cette comparaison s'il trouve que **str1** et **str3** identiques donc c'est **DEC** il va **decrementer** en premier. Il va obtenir la valeur de secteur 1 et **block 4** c'est la valeur de solde et en utilisant **Mefari_GetValue** :

```

if( strcmp(str1,str3)==0) { // if str1 = DEC
    int decr = atoi( array[1]);
    Serial.print(decr);
    // In this sample we use the second sector,
    // that is: sector #1, covering block #4 up to and including block #7
    byte sector = 1;

    byte valueBlockA = 4;
    // byte valueBlockB = 6;
    byte trailerBlock = 7;
    MFRC522::StatusCode status;
    byte buffer[18];
    byte size = sizeof(buffer);
    int32_t value;
    // Authenticate using key A
    Serial.println(F("Authenticating using key A..."));
    status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, trailerBlock, &key, &(mfrc522.uid));
    if (status != MFRC522::STATUS_OK) {
        Serial.print(F("PCD_Authenticate() failed: "));
        Serial.println(mfrc522.GetStatusCodeName(status));
        return;
    }

    status = mfrc522.MIFARE_GetValue(valueBlockA, &value);
    if (status != MFRC522::STATUS_OK) {
        Serial.print(F("mifare_GetValue() failed: "));
        Serial.println(mfrc522.GetStatusCodeName(status));
        return;
    }
    Serial.print("solde");
    Serial.print(valueBlockA);
    Serial.print("=");
    Serial.print(value);
    Serial.println("=tx19");
}
// We need a sector trailer that defines blocks 5 and 6 as Value Blocks and enables key B

```

Après il va utiliser **décrémenter** puisque dans ce cas **str1=DEC** pour décrémente et il va utiliser **MIFARE_Transfer** puisque **MIFARE_decrement** stocke seulement la valeur dans la mémoire volatile :

```

status = mfrc522.MIFARE_Decrement(valueBlockA, decr);
if (status != MFRC522::STATUS_OK) {
    Serial.print(F("MIFARE_Decrement() failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
}

status = mfrc522.MIFARE_Transfer(valueBlockA);
if (status != MFRC522::STATUS_OK) {
    Serial.print(F("MIFARE_Transfer() failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
}

// Show the new value of valueBlockB
status = mfrc522.MIFARE_GetValue(valueBlockA, &value);
if (status != MFRC522::STATUS_OK) {
    Serial.print(F("mifare_GetValue() failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
}

Serial.print("solde");
Serial.print(valueBlockA);
Serial.print("=");
Serial.print(value);
Serial.println("=tx19");

mfrc522.PCD_StopCrypto1();
}
else{
    mfrc522.PCD_StopCrypto1();
}
}

```

1-2-Sur Eclipse :

- Dans Eclipse on va lancer la communication avec **serialPort** :

```
System.out.println("f");
SerialPort[] list= SerialPort.getCommPorts();
sp =SerialPort.getCommPorts()[0];
sp.setComPortParameters(9600, Byte.SIZE, SerialPort.ONE_STOP_BIT, SerialPort.NO_PARITY);
sp.setComPortTimeouts(SerialPort.TIMEOUT_WRITE_BLOCKING, 0, 0);

if(sp.openPort()) {
    System.out.println("port is opened : ");
}
else {
    System.out.println("port is not opened :");
    return;
}
Serial_EventBasedReading(sp);
```

port COM3

S'il clique sur le bouton **incrémenter** :

```
if (e.getSource() == setButton1) {
    boolean isNumeric = creditField.getText().matches("[+-]?\\d*(\\.\\d+)?");
    if(isNumeric) {
        soldeTextField.setText("");
        Thread thread = new Thread() {
            @Override public void run() {
                try { Thread.sleep(100);}catch(Exception e) {}
                PrintWriter output = new PrintWriter(sp.getOutputStream());
                output.print("INC,"+creditField.getText());
                output.flush();
                creditField.setText("");
            }
        };
        thread.start();
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e2) {
            // TODO Auto-generated catch block
            e2.printStackTrace();
        }
    }
    else {
        JOptionPane.showMessageDialog(this, "entrer une numero");
    }
}
```

si il clique sur la button increment

OutputStream de SerialPORT

INC

S'il clique sur **décrémenter** :

```
if(e.getSource()==setButton2) { ← si il clique sur la button decrement
    boolean isNumeric = debitField.getText().matches("[+-]?\\d*(\\.\\d+)?");
    if(isNumeric) {
        soldeTextField.setText("");
        Thread thread = new Thread() {
            @Override public void run() {
                try { Thread.sleep(100);}catch(Exception e) {}
                PrintWriter output = new PrintWriter(sp.getOutputStream());
                output.print("DEC,"+debitField.getText());
                output.flush();
                debitField.setText("");
            }
        };
        thread.start();
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e2) {
            // TODO Auto-generated catch block
            e2.printStackTrace();
        }

        }else {
            JOptionPane.showMessageDialog(this, "entrer une numero");
        }
    }
}
```

DEC

on va print sur OutputStream de Serial Port

Deuxième Application

La deuxième application consiste à utiliser la Smart Card pour **stocker et embarquer un mot de passe**.

L'application vérifie d'abord l'identité de l'utilisateur : **Login/MDP**. Le mot de passe est stocké dans un bloc de la Smart Card.

Une fois authentifié, l'utilisateur pourra **demande le changement du MDP**. Celui-ci sera invité à taper le nouveau mot de passe puis le retaper une seconde fois pour vérification puis valider. C'est ce nouveau mot de passe qui devra être utilisé pour toute connexion ultérieure.

2-1- Sur Arduino :

La première chose qu'il va faire c'est de vérifier si la carte contient le mot de passe ou pas à travers la lecture du **block 14** utilisant **MFRC522_READ** après d'authentifier sur ce block 14 :

```
Serial.print(" ");
//check si la premiere fois l'utilisation de la carte
byte len = 16;
status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, 14, &key, &(mfrc522.uid)); //line 834
if (status != MFRC522::STATUS_OK) {
    Serial.print(F("Authentication failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
}

byte buffer4[18];
status = mfrc522.MIFARE_Read(14, buffer4, &len);
if (status != MFRC522::STATUS_OK) {
    Serial.print(F("Reading failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
}

Serial.write("0xchk//");
for (uint8_t i = 0; i < 16; i++) {
    Serial.write(buffer4[i] );
}
Serial.write("//0xchk");
char chk2[20]="on#";
```

block 14

if block 14 == On donc c'est pas la premiere fois d'utilisation de la carte

Apres il va lire le **block 13** utilisant **MIFARE_READ**. Le **block 12** c'est le block qui contient l'**identifiant** pour l'utiliser dans la vérification :

```
status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, 12, &key, &(mfrc522.uid)); //line 834
if (status != MFRC522::STATUS_OK) {
    Serial.print(F("Authentication failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
}

status = mfrc522.MIFARE_Read(12, buffer2, &len);
if (status != MFRC522::STATUS_OK) {
    Serial.print(F("Reading failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
}
```

read le block12 == > identifiant

```
status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, 13, &key, &(mfrc522.uid)); //line 834
if (status != MFRC522::STATUS_OK) {
    Serial.print(F("Authentication failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
}

byte buffer3[18];
status = mfrc522.MIFARE_Read(13, buffer3, &len);
if (status != MFRC522::STATUS_OK) {
    Serial.print(F("Reading failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
}
```

Maintenant si **str1** = "CHN" i.e.changer le mot de passe : il va lire le mot de passe écrit par l'utilisateur :

```
if( strcmp(str1,str5)==0) { //changer le mot de pass str1== CHN
    byte block;
    MFRC522::StatusCode status;
    byte buffer[34];
    // String login = array[1];
    String pass= array[1];
```

recupèrent le mot de pass
qu'on a obten a travers Serial.read()

Et il va écrire ce nouveau mot de passe dans **block 13** en utilisant **MIFARE_write** :

```
    //Serial.println(F("MIFARE_write() success: "));
    //changer le mot de pass str1== CHN
    int n2 = pass.length();
    char char_arr2[n2 + 1];
    strcpy(buffer, pass.c_str());
    block = 13;
    //Serial.println(F("Authenticating using key A..."));
    status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, block, &key, &(mfrc522.uid));
    if (status != MFRC522::STATUS_OK) {
        Serial.print(F("PCD_Authenticate() failed: "));
        Serial.println(mfrc522.GetStatusCodeName(status));
        return;
    }
    else Serial.println(F("PCD_Authenticate() success: "));

    // Write block
    status = mfrc522.MIFARE_Write(block, buffer, 16);
    if (status != MFRC522::STATUS_OK) {
        Serial.print(F("MIFARE_Write() failed: "));
        Serial.println(mfrc522.GetStatusCodeName(status));
        return;
    }
    else Serial.println(F("MIFARE_Write() success: "));

    }else{
        mfrc522.PCD_StopCrypto1();
    }
}
```

block13

2-2-Sur Eclipse :

Il va utiliser Eclipse pour d'abord lancer la communication avec le **SerialPort** :

```
System.out.println("f");
SerialPort[] list= SerialPort.getCommPorts();
sp =SerialPort.getCommPorts()[0];
sp.setComPortParameters(9600, Byte.SIZE, SerialPort.ONE_STOP_BIT, SerialPort.NO_PARITY);
sp.setComPortTimeouts(SerialPort.TIMEOUT_WRITE_BLOCKING, 0, 0);

if(sp.openPort()) {
    System.out.println("port is opened : ");

}else {
    System.out.println("port is not opened :");
    return;
}
Serial_EventBasedReading(sp);
```

port COM3

Ensuite il va attendre, s'il obtient **success** apres l'authentificato in il va s'authentifier, si il obtient **failed** il ne va pas pemettre l'utilisateur a s'authentifier.

```

LoginForm.java  IOException.cia  ComptForm.java  BUILTINCCLASSLOA  Main.java

public void serialEvent(SerialPortEvent ev) {
    // TODO Auto-generated method stub
    if(ev.getEventType()==SerialPort.LISTENING_EVENT_DATA_RECEIVED) {
        byte[] newData = ev.getReceivedData();
        for(int i=0;i<newData.length;i++) {
            dataBuffer += (char) newData[i];
        }
        dataBuffer2 += "\n" + dataBuffer;

        if(dataBuffer.contains("0xsuccessx0")) {
            System.out.println("success =====");
            sp.closePort();
            MenuForm cf = new MenuForm();
            cf.setTitle("compte Form");
            cf.setVisible(true);
            cf.setBounds(10,10,600,600);
            cf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            cf.setResizable(false);
            cf.show();
            dispose();
        }else if(dataBuffer.contains("0xfailedx0")) {
            System.out.println("Failed =====");
            UIManager um = new UIManager();
            um.put("OptionPane.messageForeground", Color.black);
            um.put("Panel.background", Color.white);
            um.put("OptionPane.background", Color.white);
            um.put("OptionPane.okButtonColor", Color.red);
            JButton button = new JButton("OK");
            button.setBackground(Color.BLACK);
            button.setForeground(Color.WHITE);
            button.addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent actionEvent) {
                    JOptionPane.getRootFrame().dispose();
                }
            })
        }
    }
}

```

Ici il va envoyer le login et mot de passe pour les vérifier :

```

}else {
    Thread thread = new Thread() {
        @Override public void run() {
            try { Thread.sleep(100);}catch(Exception e) {}
            PrintWriter output = new PrintWriter(sp.getOutputStream());
            output.print("LOG,"+userTextField.getText()+"#,"+passwordField.getText()+"#");
            output.flush();
        }
    };
    thread.start();
    try {
        Thread.sleep(1000);
    } catch (InterruptedException e2) {
        // TODO Auto-generated catch block
        e2.printStackTrace();
    }
}

```

LOG print to OutputStream de SerialPort

Ici il va envoyer **CHN** et le nouveau mot de passe :

```
}else {
Thread thread = new Thread() {
    @Override public void run() {
        try { Thread.sleep(100);}catch(Exception e) {}
        PrintWriter output = new PrintWriter(sp.getOutputStream());
        output.print("CHN,"+mdpTextField.getText()+"#");
        output.flush();

        dispose();
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        sp.closePort();
    }
};
thread.start();
try {
    Thread.sleep(1000);
} catch (InterruptedException e2) {
    // TODO Auto-generated catch block
    e2.printStackTrace();
}
}
```



CHN pour changer
le mot de pass

Méthodes Utilisées

◆ PICC_IsNewCardPresent()

bool MFRC522::PICC_IsNewCardPresent ()

virtuel

Renvoie true si un PICC répond à PICC_CMD_REQA. Seules les « nouvelles » cartes en état IDLE sont invitées. Les cartes de couchage dans l'état HALT sont ignorées.

Retourne
bool

◆ PCD_Authenticate()

```
MFRC522::StatusCode MFRC522::PCD_Authenticate ( octet      commande,
                                                octet      blockAddr,
                                                MIFARE_Key * clé,
                                                Uid *      uid
                                                )
```

Exécute la commande **MFRC522** MFAuthent. Cette commande gère l'authentification MIFARE pour permettre une communication sécurisée avec n'importe quelle carte MIFARE Mini, MIFARE 1K et MIFARE 4K. L'authentification est décrite dans la section 10.3.1.9 de la fiche technique **du MFRC522** et http://www.nxp.com/documents/data_sheet/MF1S503x.pdf la section 10.1. À utiliser avec les PICC MIFARE Classic. Le PICC doit être sélectionné - c'est-à-dire à l'état ACTIF(*) - avant d'appeler cette fonction. N'oubliez pas d'appeler **PCD_StopCrypto1()** après avoir communiqué avec le PICC authentifié - sinon aucune nouvelle communication ne peut démarrer.

Toutes les clés sont définies sur FFFFFFFFh lors de la livraison de la puce.

Retourne
STATUS_OK sur le succès, STATUS_??? autrement. Probablement STATUS_TIMEOUT si vous fournissez la mauvaise clé.

Paramètres

commander PICC_CMD_MF_AUTH_KEY_A ou PICC_CMD_MF_AUTH_KEY_B
blockAddr Numéro de bloc. Voir la numérotation dans les commentaires dans le fichier .h.
clé Pointeur vers la clé Crypto1 à utiliser (6 octets)
uid Pointeur vers **Uid** struct. Les 4 premiers octets de l'UID sont utilisés.

◆ MIFARE_Increment()

```
MFRC522::StatusCode MFRC522::MIFARE_Increment ( octet  blockAddr,
                                                int32_t delta
                                                )
```

MIFARE Increment ajoute le delta à la valeur du bloc adressé et stocke le résultat dans une mémoire volatile. Pour MIFARE Classic uniquement. Le secteur contenant le bloc doit être authentifié avant d'appeler cette fonction. Uniquement pour les blocs en mode « bloc de valeur », c'est-à-dire avec des bits d'accès [C1 C2 C3] = [110] ou [001]. Utilisez **MIFARE_Transfer()** pour stocker le résultat dans un bloc.

Retourne
STATUS_OK sur le succès, STATUS_??? autrement.

Paramètres

blockAddr Numéro de bloc (0-0xff).
delta Ce nombre est ajouté à la valeur de blockAddr.

◆ MIFARE_Decrement()

```
MFRC522::StatusCode MFRC522::MIFARE_Decrement ( octet  blockAddr,  
                                                int32_t delta  
                                                )
```

MIFARE Decrement soustrait le delta de la valeur du bloc adressé et stocke le résultat dans une mémoire volatile. Pour MIFARE Classic uniquement. Le secteur contenant le bloc doit être authentifié avant d'appeler cette fonction. Uniquement pour les blocs en mode « bloc de valeur », c'est-à-dire avec des bits d'accès [C1 C2 C3] = [110] ou [001]. Utilisez [MIFARE_Transfer\(\)](#) pour stocker le résultat dans un bloc.

Retourne

STATUS_OK sur le succès, STATUS_??? autrement.

Paramètres

blockAddr Numéro de bloc (0-0xff).

delta Ce nombre est soustrait de la valeur de blockAddr.

◆ MIFARE_Transfer()

```
MFRC522::StatusCode MFRC522::MIFARE_Transfer ( octet blockAddr )
```

MIFARE Transfer écrit la valeur stockée dans la mémoire volatile dans un bloc MIFARE Classic. Pour MIFARE Classic uniquement. Le secteur contenant le bloc doit être authentifié avant d'appeler cette fonction. Uniquement pour les blocs en mode « bloc de valeur », c'est-à-dire avec des bits d'accès [C1 C2 C3] = [110] ou [001].

Retourne

STATUS_OK sur le succès, STATUS_??? autrement.

Paramètres

blockAddr Numéro de bloc (0-0xff).

◆ MIFARE_Write()

```
MFRC522::StatusCode MFRC522::MIFARE_Write ( octet  blockAddr,  
                                             octet * tampon,  
                                             octet  bufferTaize  
                                             )
```

Écrit 16 octets dans le PICC actif.

Pour MIFARE Classic, le secteur contenant le bloc doit être authentifié avant d'appeler cette fonction.

Pour MIFARE Ultralight, l'opération est appelée « COMPATIBILITY WRITE ». Même si 16 octets sont transférés vers l'Ultralight PICC, seuls les 4 octets les moins significatifs (octets 0 à 3) sont écrits à l'adresse spécifiée. Il est recommandé de définir les octets restants 04h sur 0Fh sur toute la logique 0.

• Retourne

STATUS_OK sur le succès, STATUS_??? autrement.

Paramètres

blockAddr MIFARE Classic : Le numéro de bloc (0-0xff). MIFARE Ultralight: La page (2-15) à laquelle écrire.

tampon Les 16 octets à écrire dans le PICC

bufferTaize La taille de la mémoire tampon doit être d'au moins 16 octets. Exactement 16 octets sont écrits.

◆ MIFARE_Read()

```
MFRC522::StatusCode MFRC522::MIFARE_Read ( octet  blockAddr,  
                                             octet * tampon,  
                                             octet * bufferTaize  
                                             )
```

Lit 16 octets (+ 2 octets CRC_A) à partir du PICC actif.

Pour MIFARE Classic, le secteur contenant le bloc doit être authentifié avant d'appeler cette fonction.

Pour MIFARE Ultralight, seules les adresses 00h à 0Fh sont décodées. Le MF0ICU1 renvoie un NAK pour les adresses supérieures. Le MF0ICU1 répond à la commande READ en envoyant 16 octets à partir de l'adresse de page définie par l'argument de commande. Par exemple; si blockAddr est 03h alors les pages 03h, 04h, 05h, 06h sont renvoyées. Un rollback-back est implémenté : si blockAddr est 0Eh, le contenu des pages 0Eh, 0Fh, 00h et 01h est renvoyé.

La mémoire tampon doit être d'au moins 18 octets car une CRC_A est également renvoyée. Vérifie la CRC_A avant de retourner STATUS_OK.

Retourne

STATUS_OK sur le succès, STATUS_??? autrement.

Paramètres

blockAddr MIFARE Classic : Le numéro de bloc (0-0xff). MIFARE Ultralight : Première page à partir de laquelle renvoyer des données.

tampon La mémoire tampon dans laquelle stocker les données

bufferTaize Taille de la mémoire tampon, au moins 18 octets. Nombre également d'octets renvoyés si STATUS_OK.