```
// declaration des variables
enum PCD_Register : byte {
	// Page 0: Command and status
	//				0x00		// reserved for future use
	CommandReg			= 0x01 << 1,	// starts and stops command execution
	ComIEnReg			= 0x02 << 1,	// enable and disable interrupt request control bits
	DivIEnReg			= 0x03 << 1,	// enable and disable interrupt request control bits
	ComIrqReg			= 0x04 << 1,	// interrupt request bits
	DivIrqReg			= 0x05 << 1,	// interrupt request bits
	ErrorReg			= 0x06 << 1,	// error bits showing the error status of the last command executed
	Status1Reg			= 0x07 << 1,	// communication status bits
	Status2Reg			= 0x08 << 1,	// receiver and transmitter status bits
	FIFODataReg			= 0x09 << 1,	// input and output of 64 byte FIFO buffer
	FIFOLevelReg			= 0x0A << 1,	// number of bytes stored in the FIFO buffer
	WaterLevelReg			= 0x0B << 1,	// level for FIFO underflow and overflow warning
	ControlReg			= 0x0C << 1,	// miscellaneous control registers
	BitFramingReg			= 0x0D << 1,	// adjustments for bit-oriented frames
	CollReg				= 0x0E << 1,	// bit position of the first bit-collision detected on the RF interface
	//				0x0F		// reserved for future use

	// Page 1: Command
	//				0x10		// reserved for future use
	ModeReg				= 0x11 << 1,	// defines general modes for transmitting and receiving
	TxModeReg			= 0x12 << 1,	// defines transmission data rate and framing
	RxModeReg			= 0x13 << 1,	// defines reception data rate and framing
	TxControlReg			= 0x14 << 1,	// controls the logical behavior of the antenna driver pins TX1 and TX2
	TxASKReg			= 0x15 << 1,	// controls the setting of the transmission modulation
	TxSelReg			= 0x16 << 1,	// selects the internal sources for the antenna driver
	RxSelReg			= 0x17 << 1,	// selects internal receiver settings
	RxThresholdReg			= 0x18 << 1,	// selects thresholds for the bit decoder
	DemodReg			= 0x19 << 1,	// defines demodulator settings
	//				0x1A		// reserved for future use
	//				0x1B		// reserved for future use
	MfTxReg				= 0x1C << 1,	// controls some MIFARE communication transmit parameters
	MfRxReg				= 0x1D << 1,	// controls some MIFARE communication receive parameters
	//				0x1E		// reserved for future use
	SerialSpeedReg			= 0x1F << 1,	// selects the speed of the serial UART interface

	// Page 2: Configuration
	//				0x20		// reserved for future use
	CRCResultRegH			= 0x21 << 1,	// shows the MSB and LSB values of the CRC calculation
	CRCResultRegL			= 0x22 << 1,
	//				0x23		// reserved for future use
	ModWidthReg			= 0x24 << 1,	// controls the ModWidth setting?
	//				0x25		// reserved for future use
```

```
        RFCfgReg                = 0x26 << 1,   // configures the receiver gain
        GsNReg                  = 0x27 << 1,   // selects the conductance of the antenna driver
pins TX1 and TX2 for modulation
        CWGsPReg                = 0x28 << 1,   // defines the conductance of the p-driver output during
periods of no modulation
        ModGsPReg               = 0x29 << 1,   // defines the conductance of the p-driver output during
periods of modulation
        TModeReg                = 0x2A << 1,  // defines settings for the internal timer
        TPrescalerReg           = 0x2B << 1,  // the lower 8 bits of the TPrescaler value. The 4 high bits are
in TModeReg.
        TReloadRegH             = 0x2C << 1,  // defines the 16-bit timer reload value
        TReloadRegL             = 0x2D << 1,
        TCounterValueRegH       = 0x2E << 1,  // shows the 16-bit timer value
        TCounterValueRegL       = 0x2F << 1,

        // Page 3: Test Registers
        //                      0x30            // reserved for future use
        TestSel1Reg             = 0x31 << 1,   // general test signal configuration
        TestSel2Reg             = 0x32 << 1,   // general test signal configuration
        TestPinEnReg            = 0x33 << 1,   // enables pin output driver on pins D1 to D7
        TestPinValueReg         = 0x34 << 1,   // defines the values for D1 to D7 when it is used as an
I/O bus
        TestBusReg              = 0x35 << 1,   // shows the status of the internal test bus
        AutoTestReg             = 0x36 << 1,   // controls the digital self-test
        VersionReg              = 0x37 << 1,   // shows the software version
        AnalogTestReg           = 0x38 << 1,   // controls the pins AUX1 and AUX2
        TestDAC1Reg             = 0x39 << 1,   // defines the test value for TestDAC1
        TestDAC2Reg             = 0x3A << 1,  // defines the test value for TestDAC2
        TestADCReg              = 0x3B << 1         // shows the value of ADC I and Q channels
        //                      0x3C            // reserved for production tests
        //                      0x3D            // reserved for production tests
        //                      0x3E            // reserved for production tests
        //                      0x3F            // reserved for production tests
    };

    // MFRC522 commands. Described in chapter 10 of the datasheet.
    enum PCD_Command : byte {
        PCD_Idle                = 0x00,         // no action, cancels current command execution
        PCD_Mem                 = 0x01,         // stores 25 bytes into the internal buffer
        PCD_GenerateRandomID    = 0x02,         // generates a 10-byte random ID number
        PCD_CalcCRC             = 0x03,         // activates the CRC coprocessor or
performs a self-test
        PCD_Transmit            = 0x04,         // transmits data from the FIFO buffer
        PCD_NoCmdChange         = 0x07,         // no command change, can be used to modify
the CommandReg register bits without affecting the command, for example, the PowerDown bit
        PCD_Receive             = 0x08,         // activates the receiver circuits
        PCD_Transceive          = 0x0C,         // transmits data from FIFO buffer to antenna
and automatically activates the receiver after transmission
        PCD_MFAuthent           = 0x0E,         // performs the MIFARE standard authentication
as a reader
        PCD_SoftReset           = 0x0F          // resets the MFRC522
    };
```

```
// dans le loop ou le setup

mfrc522.PCD_Reset();

// 2. Clear the internal buffer by writing 25 bytes of 00h
byte ZEROES[25] = {0x00};
mfrc522.PCD_WriteRegister(FIFOLevelReg, 0x80);           // flush the FIFO buffer
mfrc522.PCD_WriteRegister(FIFODataReg, 25, ZEROES);  // write 25 bytes of 00h to FIFO
mfrc522.PCD_WriteRegister(CommandReg, PCD_Mem);              // transfer to internal buffer
```