Programmation cartes à puces

Loïc Demange
loic.demange@etud.univ-paris8.fr

avec les notes de cours de Philippe Guillot

31/4 février 2021

Pratique

Pour compiler, programmer, et tester la carte.

Dans le terminal, dans le dossier du projet

- **make** pour compiler
- **make progcarte** pour programmer (la carte dans le programmateur)
- **scat hello.script** pour tester (la carte dans le lecteur)

TP

Tester la carte avec SCAT, et rajouter des lignes dans hello.script pour tester la fonction sortir_data().

EEPROM

L'EEPROM est une mémoire non volatile inscriptible par le programme.

Pour pouvoir l'utiliser, il faut

- inclure la bibliothèque avec **#include<avr/eeprom.h>**.
- déclarer les variables concernées avec le tag **EEMEM**.
- utiliser les différentes fonctions permettant d'écrire et de lire la mémoire EEPROM.

Remarque On nomme les variables avec le préfixe **ee_** pour simplifier la lecture.

```
uint8_t ee_val EEMEM = 0;
```

EEPROM

```
// Permet de lire un entier de 8 octets dans l'EEPROM
uint8_t eeprom_read_byte(uint8_t * dest);

// Permet de lire un mot (entier de 16 octets) dans l'EEPROM
uint16_t eeprom_read_word(uint16_t * dest);

// Permet de lire n octets de données dans l'EEPROM
void eeprom_read_block(void * dest, void * src, uint8_t n);

// Permet d'ecrire un entier de 8 octets dans l'EEPROM
void eeprom_write_byte(uint8_t * dest, uint8_t val);

// Permet d'ecrire un mot (entier de 16 octets) dans l'EEPROM
void eeprom_write_word(uint16_t * dest, uint16_t val);

// Permet d'ecrire n octets de données dans l'EEPROM
void eeprom_write_block(void * src, void * dest, uint8_t n);
```

TP

Modifier les fonctions intro_data() et sortir_data() pour que les données soient conservées dans l'EEPROM.

Projet porte monnaie

Le but du projet est de réaliser un porte monnaie électronique, avec pour fonctionnalités

- introduction des données personnelles
- lecture des données personnelles
- lecture du solde
- dépôt d'argent
- retrait d'argent

Le solde sera sur 16 bits, et stocké dans un entier non signé (**uint16_t**).
Les données personnelles et le solde seront évidemment dans une mémoire non volatile.
La classe de la carte sera la classe libre **0x81**.

Projet porte monnaie

Deux problèmatiques à prendre en compte

- l'Endianness
- les dépôts ou retraits trop importants

Projet porte monnaie

L'Endianness est la manière d'organiser les octets en mémoire.

- Le Big Endian est le fait de stocker l'octet de poids fort en premier (de gauche à droite).
- Le Little Endian est le fait de stocker l'octet de poids faible en premier (de droite à gauche).

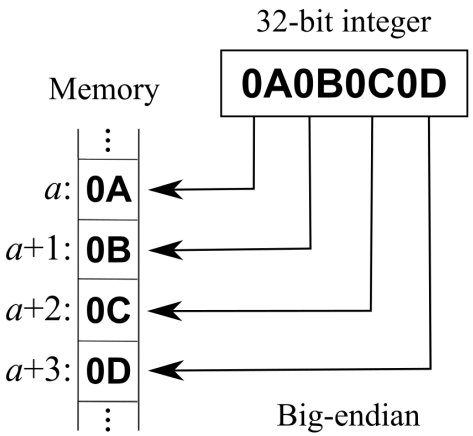

Figure: Big Endian (wikipédia)

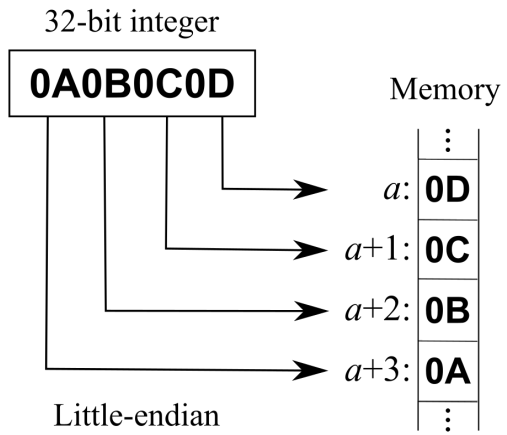

Figure: Little Endian (wikipédia)

Projet porte monnaie

Le solde est stocké sur deux octets. Or, on ne peut envoyer et recevoir qu'octet par octet.

Il faudra donc savoir découper un entier de 16 bits en deux entiers de 8 bits, et inversement reformer un entier de 16 bits avec deux entiers de 8 bits.

L'Endianness est donc important à prendre en compte.

Projet porte monnaie

Le solde est stocké comme un entier non signé, il faut donc faire attention aux dépôts et aux retraits invalides.

- Si on essaye de retirer une somme trop importante, notre solde va boucler ($0 - 1 = 65535$).
- Si on essaye de déposer une somme trop importante, notre solde va boucler ($65535 + 1 = 0$).