



Electronics Project

Day 05 : EEPROM

contact@42chips.fr

Résumé: Because writing too many times at the same spot is what made cars crash

Chapitre I

Préambule

Une EEPROM (Electrically Erasable Programmable Read-Only Memory) est un type de mémoire non-volatile qui peut être utilisée pour stocker des données sur un microcontrôleur.

Il peut être écrit et effacé plusieurs fois et conserve ses données lorsque l'alimentation est coupée.

Les EEPROM sont utiles pour stocker des données qui doivent être conservées même lorsque le microcontrôleur n'est pas alimenté, telles que des paramètres de configuration ou des données d'étalonnage.

Ils sont plus lents et ont une capacité moindre que d'autres types de mémoire, tels que la SRAM ou la mémoire flash, mais sont toujours utiles dans une variété d'applications.

Chapitre II


General instructions

Unless explicitly stated otherwise, the following instructions will be valid for all assignments.

- The language used for this project is C.
- It is not necessary to code according to the 42 norm.
- The exercises are ordered very precisely from the simplest to the most complex. Under no circumstances will we consider or evaluate a complex exercise if a simpler one is not perfectly successful.
- You must not leave any files other than those explicitly specified by the exercise instructions in your directory during peer evaluation.
- All technical answers to your questions can be found in the **datasheets** or on the Internet. It is up to you to use and abuse these resources to understand how to complete your exercise.
- You must use the datasheet of the microcontroller provided to you and comment on the important parts of your program by indicating where you found the clues in the document, and if necessary, explaining your approach. Don't write long blocks of text, keep it clear.
- Do you have a question? Ask your neighbor to the right or left. You can ask in the dedicated channel on the Piscine's Discord, or as a last resort, ask a staff member.

Chapitre III

Tutoriel

	Exercice : 00
Mémorisation d'état	
Dossier de rendu : <i>ex00/</i>	
Fichiers à rendre : <code>Makefile</code> , <code>main.c</code>	
Fonctions Autorisées : <code>avr/io.h</code> , <code>avr/eeprom.h</code>	

- utiliser l'EEPROM pour sauvegarder et restaurer l'état d'un compteur.
- utiliser un bouton pour incrémenter le compteur.
- utiliser les LEDs sur la carte pour afficher l'état actuel du compteur.



Ne supposez pas que l'EEPROM est initialisée à 0 lorsque vous n'avez pas encore écrit dedans.



Exercice : 01

Multiplexeur

Dossier de rendu : *ex01/*Fichiers à rendre : **Makefile**, **main.c**Fonctions Autorisées : **avr/io.h**, **avr/eeprom.h**


- utiliser l'EEPROM pour sauvegarder et restaurer l'état de 4 compteurs.
- utiliser l'EEPROM pour sauvegarder et restaurer le compteur sélectionné actuel.
- utiliser le bouton SW2 pour incrémenter le compteur.
- utiliser le bouton SW3 pour sélectionner un compteur.
- utiliser les LEDs sur la carte pour afficher l'état actuel du compteur sélectionné.



Il peut être une bonne idée de regarder ce qu'est un nombre magique ;)

Chapitre IV

Grind avant le boss final

	Exercice : 02
The only thing I know for real	
Dossier de rendu : <i>ex02/</i>	
Fichiers à rendre : <i>Makefile</i> , <i>main.c</i>	
Fonctions Autorisées : <i>avr/io.h</i> , <i>avr/eeprom.h</i>	


Pour cet exercice, vous devez écrire un nombre magique devant le bloc de mémoire que vous écrivez dans l'EEPROM. Cela est nécessaire pour savoir que ces données ont déjà été écrites par vous. Ne réécrivez pas les octets du bloc de données si la valeur est identique.

En gardant cela à l'esprit, écrivez les fonctions suivantes :

```
bool safe_eeprom_read(void *buffer, size_t offset, size_t length);  
bool i2c_start(void * buffer, size_t offset, size_t length)
```



Le nombre de cycles d'écriture sur une EEPROM est limité, mais pas le nombre de lectures.

	Exercice : 03
EEMPROMalloc	
Dossier de rendu : <i>ex03/</i>	
Fichiers à rendre : <i>Makefile</i> , <i>main.c</i>	
Fonctions Autorisées : <i>avr/io.h</i> , <i>avr/eeprom.h</i>	

Vous devez maintenant écrire 3 fonctions qui vous permettront, comme un dictionnaire, d'écrire, de réserver et de libérer des espaces mémoire dans l'EEPROM.

Vous devez nettoyer les espaces mémoire supprimés pour pouvoir les réallouer pour des appels futurs.

```
bool eepromalloc_write(uint16_t id, void *buffer, uint16_t length)
bool eepromalloc_read(uint16_t id, void *buffer, uint16_t length)
bool eepromalloc_free(uint16_t id)
```




Vous pouvez supposer que vous êtes le seul à utiliser l'EEPROM.
Si vous n'avez plus d'espace disponible dans l'EEPROM lors de l'allocation, retournez simplement faux.



Il est également inutile de dire que vous devez ne jamais oublier ou corrompre des données qui n'ont pas été libérées.

Chapitre V

BOSS FINAL

	Exercice : 03
EEMPRoMalloc	
Dossier de rendu : <i>ex03/</i>	
Fichiers à rendre : Makefile , main.c	
Fonctions Autorisées : avr/io.h , avr/eeprom.h	

Écrire une interface en ligne de commande sur le port UART de votre microcontrôleur.

Elle peut stocker une paire de clé/valeur de chaînes qui ne peuvent pas être perdues lors du redémarrage.

Elle peut prendre 3 commandes :

- **READ** Donné une clé, récupère la valeur associée. Si non trouvée, retourne une nouvelle ligne.
- **WRITE** Donnez une clé et une valeur, tentez de les stocker et rappez sur l'état.
- **FORGET** Donnez une clé, supprime la paire clé/valeur si elle est trouvée.

```
> READ "lol"
> WRITE "lol" "je sais pas"
Done.
> READ "lol"
"je sais pas"
> FORGET "lol"
Done.
> READ "lol"
```