



Piscine embarquée

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.

Elle peut être écrite et effacée 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.

Elles sont plus lentes 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


Consignes générales

Sauf contradiction explicite, les consignes suivantes seront valables pour tous les exercices.

- Le langage utilisé pour ce projet est le C.
- Il n'est pas nécessaire de coder à la norme de 42.
- Les exercices sont très précisément ordonnés du plus simple au plus complexe. En aucun cas nous ne prendrons en compte ni n'évaluerons un exercice complexe si un exercice plus simple n'est pas parfaitement réussi.
- Vos exercices seront évalués par des responsables de l'association 42Chips.
- Vous ne devez laisser aucun autre fichier que ceux explicitement spécifiés par les énoncés des exercices dans votre répertoire lors de la peer-évaluation.
- Toutes les réponses à vos questions techniques se trouvent dans les **datasheets** ou sur Internet. À vous d'utiliser et d'abuser de ces sujets pour comprendre comment réaliser votre exercice.
- Vous devez utiliser la datasheet du microcontrôleur qui vous est fourni et commenter les parties importantes de votre programme en renseignant où vous avez trouvé les indices dans le document, et, si nécessaire, expliquer votre démarche. Ne faites pas des pavés non plus. Il faut que cela reste clair.
- Vous avez une question ? Demandez à votre voisin de droite ou de gauche. Vous pouvez demander sur le salon dédié dans le Discord de la piscine ou en dernier recours à un staff.

Chapitre III

Tutoriel

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

- Utilisez l'EEPROM pour sauvegarder et restaurer l'état d'un compteur.
- Utilisez le bouton SW1 pour incrémenter le compteur.
- Utilisez les LEDs D1, D2, D3, D4 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**, ***.c**, ***.h**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 SW1 pour incrémenter le compteur.
- Utiliser le bouton SW2 pour sélectionner un compteur.
- Utiliser les LEDs D1, D2, D3, D4 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 : *ex02/*

Fichiers à rendre : **Makefile**, ***.c**, ***.h**

Fonctions Autorisées : **avr/io.h**, **avr/eeprom.h**


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 safe_eeprom_write(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
Bobby ; DROP TABLE	
Dossier de rendu : <i>ex03/</i>	
Fichiers à rendre : Makefile, *.c, *.h	
Fonctions Autorisées : avr/io.h, avr/eeprom.h	

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 lorsque vous devez allouer de l'espace.



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

Chapitre V

BOSS FINAL

	Exercice : 04
EEPROMalloc	
Dossier de rendu : <i>ex04/</i>	
Fichiers à rendre : <i>Makefile, *.c, *.h</i>	
Fonctions Autorisées : <i>avr/io.h, avr/eeprom.h</i>	

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

Elle permet de stocker des paires de clé/valeur de chaînes de caractères qui doivent persister lors de redémarrages de votre devkit (Cela inclus éteindre le devkit). Les clés et les valeurs sont délimitées avec des guillemets (").

Elle prend 3 commandes :

- **READ** Donnez une clé, récupère la valeur associée. Si non trouvée, retourne une nouvelle ligne vide.
- **WRITE** Donnez une clé et une valeur, tentez de les stocker et donne l'état :
 - `true` : Done.
 - `false` : No space left on device.
- **FORGET** Donnez une clé, supprime la paire clé/valeur si elle est trouvée et donne l'état :
 - `true` : Done.
 - `false` : Not found.



L'exercice demandé ne vous demande pas une implémentation de malloc. Simplement une méthode pour écrire sur des bouts de mémoires sans écraser ce qui y est déjà stocké.
Ne vous compliquez pas la tâche.


```
> READ "key-1"

> WRITE "key-1" "dunno"
Done.
> READ "key-1"
"dunno"
> FORGET "key-2"
Not found.
> FORGET "key-1"
Done.
> READ "key-1"
```

```
> READ "key-1"

> WRITE "key-1" "very long string, maybe something from wikipedia ?"
Done.
> WRITE "key-2" "very long string, maybe something from wikipedia ?"
Done.
...
> WRITE "key-XXX" "very long string, maybe something from wikipedia ?"
No space left.
```