



Piscine embarquée

Module00 : Premiers programmes

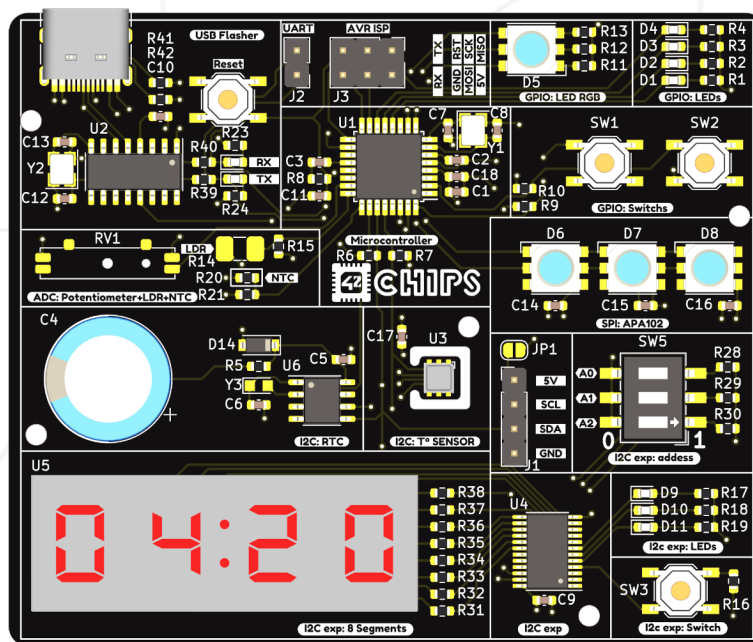
contact@42chips.fr

Résumé: Il était une fois un petit spectre...

Version: 1

Chapitre I

Préambule



Pour l'ensemble de la piscine, vous utiliserez ce kit de développement. Il est basé sur un ATmega328p, une très belle pièce pleine de ressources, tout à fait dans l'air du temps.

Les exercices sont à réaliser en C. Vous devrez les cross-compiler pour l'architecture AVR avec `avr-gcc`.

Pour écrire votre binaire sur la mémoire flash du kit, vous devez utiliser `avr-objcopy` et `avrdude`. Vous devrez utiliser le programmeur "`arduino`", avec un baud rate de 115200.

Les 2 documents les plus importants durant toute cette piscine sont :

- le datasheet de l'ATmega328p (653 pages de pur bonheur) ;
- le schéma du devkit.

Bon courage à tous, malgré l'intensité de la piscine, essayez de ne pas vous mettre sous tension ! Nous sommes sûrs que vous réaliserez votre plein potentiel !

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


Setup

	Exercice : 00
Makefile	
Dossier de rendu : <i>ex00/</i>	
Fichiers à rendre : Makefile , *.c , *.h	
Fonctions Autorisées : avr-gcc , avr-objcopy , avrdude	

- Le fichier **main.c** doit contenir un programme **main()** ne contenant aucune instruction.
- Écrire un fichier **Makefile** avec la règle **all** qui exécute la règle **hex** puis la règle **flash**.
- La règle **hex** compile le fichier **main.c** en **main.bin** avec une variable **F_CPU** pour sélectionner la fréquence du microcontrôleur. Ensuite, génère le fichier **main.hex** à partir du fichier **main.bin**.
- La règle **flash** copie ce fichier **main.hex** dans la flash du microcontrôleur.
- Le **Makefile** devra également implémenter la règle **clean** qui supprime les fichiers **main.hex** et **main.bin**.

Chapitre IV


Que la lumière soit !

	Exercice : 01
Une lueur d'espoir	
Dossier de rendu : <i>ex01/</i>	
Fichiers à rendre : Makefile , *.c , *.h	
Fonctions Autorisées : avr/io.h	


- Vous devez écrire un programme qui permet d'allumer la LED D1 (PB0).
- Vous devez utiliser uniquement les registres AVR (DDRX , PORTX, PINX).



Vous devez à chaque fois expliquer la fonction et les valeurs assignées aux registres en commentaire !

	Exercice : 02
Lumos	
Dossier de rendu : <i>ex02/</i>	
Fichiers à rendre : Makefile , *.c , *.h	
Fonctions Autorisées : avr/io.h , util/delay.h	

- Vous devez écrire un programme qui allume la LED D1 (PB0) lorsqu'on appuie sur le bouton SW1 (PD2).
- Lorsque le bouton est relâché, la LED s'éteint.
- Vous devez utiliser uniquement les registres AVR (DDRX, PORTX, PINX).

	Exercice : 03
Jour, nuit, jour, nuit	
Dossier de rendu : <i>ex03/</i>	
Fichiers à rendre : <code>Makefile</code> , <code>*.c</code> , <code>*.h</code>	
Fonctions Autorisées : <code>avr/io.h</code> , <code>util/delay.h</code>	


- Vous devez écrire un programme qui inverse l'état de la LED PB0 à chaque fois que le bouton SW1 (PD2) passe de l'état relâché à l'état **appuyé**.
- Vous devez utiliser uniquement les registres AVR (DDRX, PORTX, PINX).



Attention aux **effets rebond** !

Chapitre V

Bonus : C'est pas sorcier !

	Exercice : 04
Compteur binaire	
Dossier de rendu : <code>ex04/</code>	
Fichiers à rendre : <code>Makefile</code> , <code>*.c</code> , <code>*.h</code>	
Fonctions Autorisées : <code>avr/io.h</code> , <code>util/delay.h</code>	

- Vous devez écrire un programme qui :
 - chaque fois que vous pressez le bouton SW1 incrémente une valeur ;
 - chaque fois que vous pressez le bouton SW2 décrémente une valeur ;
 - et affiche en permanence cette valeur sur les LEDs D1 D2 D3 et D4 en binaire.
- Vous devez utiliser uniquement les registres AVR (DDRX, PORTX, PINX).



Si les 4 LEDs étaient sur les GPIO PB0, PB1, PB2, PB3, cet exercice serait plus simple.

Malheureusement la LED D4 est sur PB4 au lieu de PB3 car PB3 est utilisé pour autre chose.

Il va falloir manipuler des bits.