



# Electronics Project

## TP Prog 0 : Premiers programmes

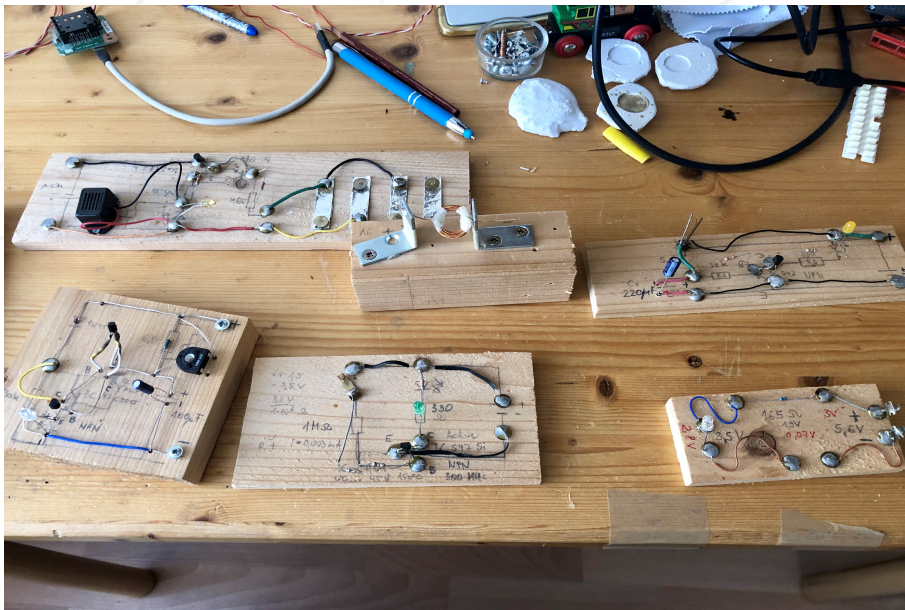
`pedago@42chips.fr`

*Résumé: Initiation à la cross-compilation et à l'utilisation des registres AVR*

# Table des matières

# Chapitre I

## Préambule



Aux débuts de la radio, les amateurs clouaient des fils de cuivre nus ou des borniers sur une planche de bois et y soudaient des composants électroniques.

Parfois, un diagramme schématique sur papier était d'abord collé sur la carte comme guide pour placer les bornes, puis les composants et les fils étaient installés aux symboles associés sur le schéma.

L'utilisation de punaises ou de petits clous comme poteaux de montage était également courante.

Les planches à pain ou breadboards ont évolué au fil du temps, le terme étant maintenant utilisé pour toutes sortes de prototypes d'appareils électroniques.

La breadboard la plus couramment utilisée aujourd'hui est généralement en plastique blanc et permet d'enficher les composants sans nécessité d'apposer de la soudure. Elle a été conçue par Ronald J. Portugal en 1971, et est maintenant utilisée couramment par tous pour prototyper des parties de projet.

# Chapitre II


## Consignes générales

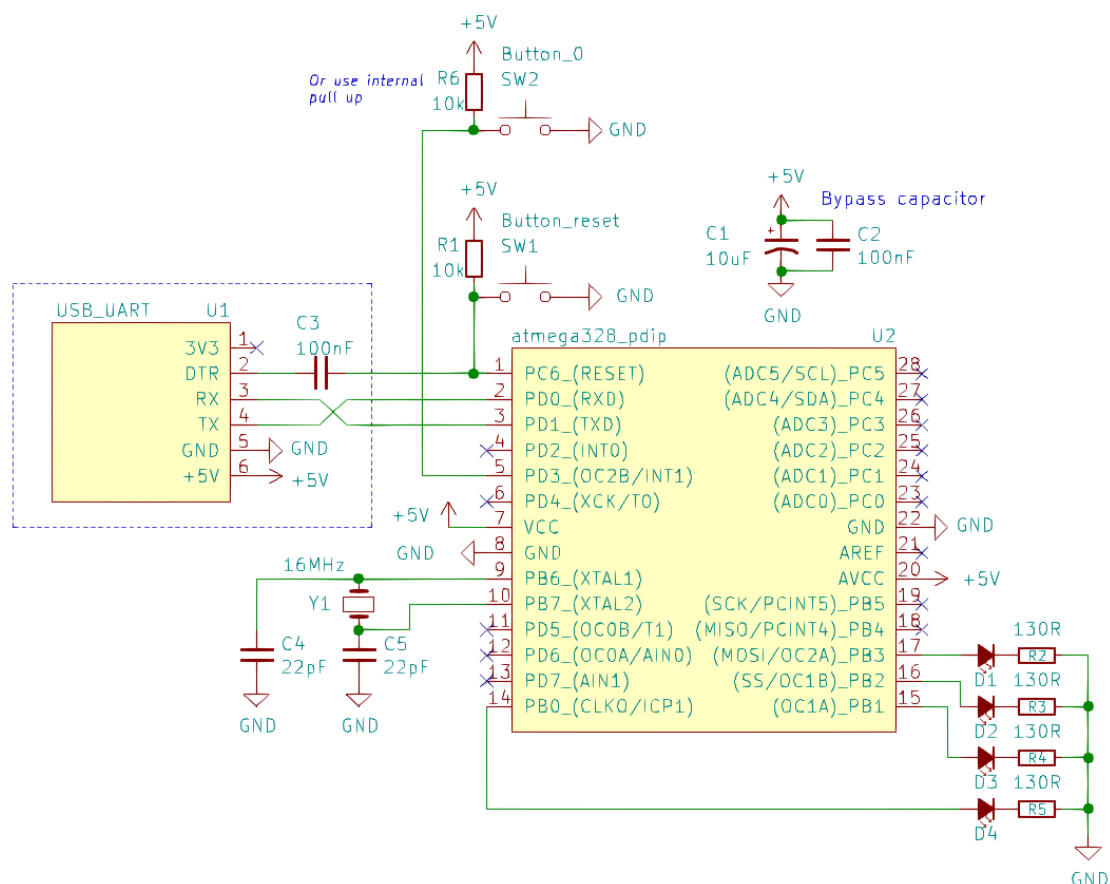
Sauf contradiction explicite, les consignes suivantes seront valables pour tous les TPs

- 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.
- Vous avez une question ? Demandez à votre voisin de droite ou de gauche. Vous pouvez demander sur le salon dédié dans le discord [42Chips](#) ou en dernier recours à un responsable 42Chips.
- Toutes les réponses à vos questions techniques se trouvent dans les **datasheets** ou sur Internet. A 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.
- Écoutez attentivement les encadrants lors des séances de TP, ils vous donneront des éléments essentiels sur le fonctionnement du microcontrôleur.

# Chapitre III

## Exercice 00 : Breadboard


	Exercice : 00
Breadboard	
Dossier de rendu : <i>ex00/</i>	
Fichiers à rendre :	
Fonctions Autorisées :	



- Utilisez les composants, breadboard, et câbles mis à votre disposition pour réaliser votre propre devkit

# Chapitre IV


## Exercice 01 : Makefile

	Exercice : 01
Makefile	
Dossier de rendu : <i>ex01/</i>	
Fichiers à rendre : <b>Makefile</b> , <b>main.c</b>	
Fonctions Autorisées : <b>avr-gcc</b> , <b>avr-objcopy</b> , <b>avrdude</b>	

- 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 V

## Exercice 02 : Une lueur d'espoir

	Exercice : 02
Une lueur d'espoir	
Dossier de rendu : <i>ex02/</i>	
Fichiers à rendre : <b>Makefile</b> , <b>main.c</b>	
Fonctions Autorisées : <b>avr/io.h</b>	


- Vous devez écrire un programme qui permet d'allumer la LED D1 (PB3).
- 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 !

# Chapitre VI

## Exercice 03 : Clignotant

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

- Vous devez écrire un programme qui permet d'allumer et éteindre la LED D1 (PB3) à une fréquence de 1Hz.
- Vous devez écrire un code qui permet d'attendre plusieurs centaines de millisecondes et qui sera inséré dans la boucle infinie du programme ( pas de TIMER hardware ).
- Le changement d'état de la LED doit être fait avec une unique **opération bitwise**, il ne faut pas utiliser de condition ( if else ).
- Vous devez utiliser uniquement les registres AVR ( DDRX , PORTX, PINX ).




1Hz == ( allumé 0.5sec et éteint 0.5sec )  
L'exercice sera considéré comme invalide si votre main return.



# Chapitre VII


## Exercice 04 : Lumos

	Exercice : 04
Lumos	
Dossier de rendu : <i>ex04/</i>	
Fichiers à rendre : <b>Makefile</b> , <b>main.c</b>	
Fonctions Autorisées : <b>avr/io.h</b>	

- Vous devez écrire un programme qui allume la LED D1 (PB3) lorsqu'on appuie sur le bouton.
- Lorsque le bouton est relâché, la LED s'éteint.
- Vous devez utiliser uniquement les registres AVR ( DDRX, PORTX, PINX ).

# Chapitre VIII

## Exercice 05 : Jour, nuit, jour, nuit

	Exercice : 05
Jour, nuit, jour, nuit	
Dossier de rendu : <i>ex05/</i>	
Fichiers à rendre : <code>Makefile</code> , <code>main.c</code>	
Fonctions Autorisées : <code>avr/io.h</code>	


- Vous devez écrire un programme qui inverse l'état de la LED D1 (PB3) à chaque fois que le bouton passe de l'état relâché à l'état **appuyé**.
- Vous devez utiliser uniquement les registres AVR ( `DDRX`, `PORTX`, `PINX` )



Attention aux **effets rebonds** !

# Chapitre IX

## Exercice 06 : Compteur binaire

	Exercice : 06
Compteur binaire	
Dossier de rendu : <i>ex06/</i>	
Fichiers à rendre : <b>Makefile</b> , <b>main.c</b>	
Fonctions Autorisées : <b>avr/io.h</b>	

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