



Electronics Project, TP Prog

TP Prog 3 : Protocole UART

[pedago@42chips.fr](mailto:pedago@42chips.fr)

*Résumé: Initiation au protocole UART*

# Table des matières

# Chapitre I

## Préambule

Le code [Morse](#) international (ou l'alphabet Morse international) est un code, qui permet de transmettre un texte à l'aide de séries d'impulsions courtes et longues, qu'elles soient produites par des signes, une lumière, un signal électrique, un son ou un geste.

Ce code est souvent attribué à **Samuel Morse**, mais plusieurs contestent cette primauté, et tendent à attribuer la paternité du langage à son assistant, **Alfred Vail**.

Inventé en 1832 pour la télégraphie, ce codage de caractères assigne à chaque lettre, chiffre et signe de ponctuation une combinaison unique de signaux intermittents. Le code morse est considéré comme le précurseur des communications numériques.

Le morse est principalement utilisé par les militaires comme moyen de transmission, souvent chiffrée, ainsi que dans le civil pour certaines émissions à caractère automatique :

- radiobalises en aviation,
- indicatif d'appel des stations maritimes,
- des émetteurs internationaux (horloges atomiques),
- ou bien encore pour la signalisation maritime par certains transpondeurs radar.

Le morse est également pratiqué par des amateurs comme :

- les radioamateurs
- les scouts (morse sonore et lumineux),
- les plongeurs et alpinistes (morse lumineux),
- par des joueurs pour résoudre des énigmes,
- ainsi que comme sonnerie par défaut de réception de message pour les téléphones portables de marque Nokia (« SMS SMS » en morse).

# 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 : write(2)


	Exercice : 00
write(2)	
Dossier de rendu : <i>ex00/</i>	
Fichiers à rendre : <b>Makefile</b> , <b>main.c</b>	
Fonctions Autorisées : <b>avr/io.h</b>	

- Le microcontrôleur AVR ATmega328P possède 1 périphérique UART que vous devez utiliser dans cet exercice pour communiquer avec un ordinateur.
- Sur le PC, pour lire le port série on utilise le programme **screen** depuis un terminal.
- Vous devez écrire une fonction **uart\_init** qui initialise l'UART.
- Une fonction **uart\_tx** qui écrit un caractère sur le port série du PC.
- l'UART du MCU doit être configuré en 115200 8N1.
- UBRRL doit être calculé en fonction de **UART\_BAUDRATE** et **F\_CPU**
- Le programme doit écrire 'Z' sur le port serie à 1Hz. (faites comme vous voulez).

```
void uart_tx(char c);
```

# Chapitre IV

## Exercice 01 : print\_str

	Exercice : 01
print_str	
Dossier de rendu : <i>ex01/</i>	
Fichiers à rendre : <b>Makefile</b> , <b>main.c</b>	
Fonctions Autorisées : <b>avr/io.h</b>	


- Vous devez écrire une fonction `uart_printstr` qui sera appelée toutes 2 secondes pour afficher **Hello World!** sur le port série.
- l'UART du MCU doit être configuré en 115200 8N1.
- La boucle infinie du programme doit rester vide.

```
void uart_printstr(const char* str);
```

```
Hello World!  
Hello World!  
Hello World!  
...
```

# Chapitre V

## Exercice 02 : read(2)


	Exercice : 02
read(2)	
Dossier de rendu : <i>ex02/</i>	
Fichiers à rendre : <b>Makefile</b> , <b>main.c</b>	
Fonctions Autorisées : <b>avr/io.h</b>	

- Maintenant vous allez devoir écrire une fonction **uart\_rx** qui attend de recevoir un caractère sur le port UART du MCU puis le retourne.
- Vous devez écrire un programme qui utilise votre fonction **uart\_rx**.
- Il doit écrire les caractères reçus depuis **uart\_rx** sur le port série avec votre fonction **uart\_tx** (ex00).

```
char uart_rx(void);
```

# Chapitre VI

## Exercice 03 : Majuscules


	Exercice : 03
Majuscules	
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 renvoie un echo sur le port série, mais en transformant les minuscules en majuscules et les majuscules en minuscules avant de les renvoyer.
- Attention, cette fois-ci au lieu d'utiliser votre `uart_rx`, vous devez utiliser une interruption pour détecter qu'un nouveau caractère est sur le port UART.
- La boucle infinie du programme doit rester vide.



# Chapitre VII

## Exercice 04 : Login

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

- Créer 2 chaînes de caractères, une username et une password.
- Afficher un prompt sur le port série du mac qui demande le username et password.
- Quand on tape le username il y a un echo.
- Quand on tape le password il y a un echo mais avec des '\*'.
- La touche **Backspace** efface un caractère.
- La touche **Enter** valide la saisie.
- Si l'username et le password sont corrects, le programme affiche **Coucou [username]** ! et fait clignoter les LEDs. (faite vous plaisir)
- Sinon le programme affiche **Mauvaise combinaison username/password** .

```
Bonjour ! Entrez votre login:
username: spectre
password: *****
Mauvaise combinaison username/password

Bonjour ! Entrez votre login:
username: spectre
password: *****
Coucou spectre !
```