SY23 Microprocesseur

# Introduction

Ce projet propose de réaliser plusieurs mini-projet sur différents microprocesseurs. La première partie consiste à mettre un programme sur un msp430 de 3 manières différentes : en le programmant directement à l’aide de fonction bash, en utilisant un logiciel dédié Energia et enfin en y installant un os qui exécute au démarrage un process avec le programme créé. La deuxième partie consiste à contrôler un bandeau d’affichage piloter par une carte FoxG20.Tout d’abord il faut faire un script bash qui envoie les données en paramètre sur le bandeau. Ensuite il faut faire un script qui récupère les données envoyées depuis une page html et les envois sur le bandeau. La dernière partie consiste à écrire un driver qui contrôle des LED et des interrupteurs.

# MSP430

## Programmation directe

### Port 8 bits avec délai

Ce petit programme très simple possède une variable de type char utilisé comme compteur. L’avantage du char en c est qu’il est toujours codé sur 8 bits ce qui permet d’avoir toujours une taille fixe qui correspond exactement au nombre de LED.

Le compteur s’incrémente chaque fois que la boucle for de la fonction delay se termine. Si le compteur arrive à la valeur max est doit s’incrémenter de nouveau, il revient à zéro car la variable char est dépassé.

J’ai également créé une fonction qui permet d’assigner plus facilement un octet au port A. (void write\_on\_PORTA(char value);)

### Port 8 bits avec interruption

Ce programme est le même que le précédent hormis que la fonction delay est remplacée par le composant timer. Celui-ci incrémente le compteur à chaque fois qu’il finit son propre comptage.

### Afficheur 7 segments

Ce programme permet deux choses commandable avec le premier interrupteur du port B :

1. Récupération et affichage de la température

Ce programme récupère la valeur de la température ambiante et l’affiche sur un double afficheur 7 segments. Le double afficheur permet d’afficher une dizaine et une unité.

Le programme utilise un timer qui à chaque time\_out, lit la valeur du capteur de température.

Un autre timer à chaque time\_out converti cette valeur, puis, sélectionne l’affichage unité ou dizaine (chaque time\_out l’affichage alterne entre les deux). La valeur d’unité ou de dizaine codé sur un char est ensuite converti en un nombre graphique et envoyé sur le port A correspondant au 7 segments.

1. Chronomètre de 0 à 99 de période 1 seconde avec affichage

Ce chronomètre fonctionne de la même manière que les deux programmes avec LED précédents, à chaque time out d’un timer de période 1 seconde le chronomètre s’incrémente. L’affichage est piloté de la même manière que pour la température : un timer très rapide affiche soit les dizaines soit les unités de manière à ce que l’œil nue à l’impression que l’affichage des deux chiffres soit fait en même temps.

J’ai réalisé une fonction permettant de transformer une valeur décimale écrits sur un char en un chiffre visible sur le 7 segments, ainsi qu’une fonction convertissant un la température en Celsius.

J’ai utilisé un uint16\_t pour avoir également une taille fixe pour la valeur de température codé sur 16 bits.

## Energia

### Compteur 8 bits

Le programme est le même que celui de la partie Port 8 bits avec délai. Un compteur codé sur un char qui s’incrémente jusqu’à la valeur max. L’incrémentation se fait dans la boucle loop qui possède un délai de 1 seconde entre chaque tour.

Le programme utilise la classe port pour gérer le Port A.

### Affichage température sur 7 segments

Ce programme est similaire à la partie température de Afficheur 7 segments. J’ai rajouté une méthode dans la classe port qui converti et écrit sur l’afficheur la valeur passée en paramètre. Cette méthode est simplement la fonction similaire de la partie Afficheur 7 segments précédente auquel j’ai rajouté un flag pour déterminer si le nombre est à écrire est une unité ou une dizaine. Je récupère la valeur de la température grâce à la fonction *analogread(A10)*. A10 étant l’adresse du capteur de température.

Malheureusement, je n’arrive pas récupérer une bonne valeur de températures. Je ne reçois que des valeurs aberrantes. Je pense que energia converti en amont la valeur de température mais je ne connais pas cette conversion. J’ai testé à la suite le programme de la partie Afficheur 7 segments et celui-ci est les résultats lus directement par le capteur (sans passer par aucune conversion de ma part) est différent. Ce qui ne devrait pas être le cas.

## Contiki OS

### Compteur 8 bits

Ce programme fonctionne encore de la même manière que les compteur 8 bits précédents. La seule différence est que le délai est piloté par un etimer de l’OS.

### Chronomètre de 10 secondes sur 7 segments

Ce programme est similaire à la partie compteur de la partie Afficheur 7 segments. C’est un compteur de 0 à 10, qui est affiché sur le 7 segments. Le rafraîchissement se fait tous les tours d’une boucle while alors l’incrémentation se tous les 50 tour de boucles.

La différence est que le programme utilise deux interrupteurs :

* Le premier bloque le fonctionnement du chronomètre tant qu’il est enclenché.
* Le deuxième remet à 0 et bloque le fonctionnement du chronomètre tant qu’il est enclenché.

### Horloge

Ce programme est similaire au précédent sauf qu’il compte jusqu’à 60 secondes.

Je conclus cette partie en précisant que personnellement le développement en direct est bien plus pratique pour pouvoir parfaitement contrôler le microprocesseur. La programmation par application permet d’avoir un environnement de programmation plus simple qu’en direct ce qui est un avantage intéressant. Mais la programmation par energia est problématique car energia effectue certaine chose de manière « caché » ce qui donne des résultats inattendus. J’ai eu l’impression de moins bien pouvoir contrôler le microcontrôleur.

# Driver Bandeau

## Script Bash

Ce script bash très simple prend en argument le message à envoyer (pas besoin d’écrire le message entre guillemet, sauf si les caractères forme un comande linux). Le script écrit alors le message grâce à echo dans le driver du bandeau. Le message est formaté pour pouvoir être affiché sur le bandeau de gauche à droite.

Il faut taper la commande « sudo ./drivermode.sh 1 » si on veut voir le message sans la mise en forme.

## Script CGI

Le script CGI récupère un texte envoyé sur la page html, puis extrait la partie mise en forme et la partie message de ce texte. Le tout est ensuite envoyé sur le driver du bandeau en ajoutant la partie effet à « 00\V01\X» et le message entre les chevrons.

Malheureusement, je n’arrive pas à décoder les caractères spéciaux (par exemple : « é »). Je n’ai pas trouvé de solution qui permettais de faire cela. La commande iconv ne fait rien et les solutions utilisant printf ne fonctionne pas dans le script (il fonctionne seulement dans le terminal).

# Driver LED

Le driver permet de contrôler 8 leds et lire 8 interrupteurs connecté à un carte FoxG20.

Les deux fonctions importantes sont la fonction read et la focntion write.

### Read

Read lit la valeur des interrupteurs relié au port A. Il lit la valeur de chaque pin GPIO en commençant par la gauche (le MSB) puis s’arrête s’il atteint le pin 0 du port ou bien le nombre max de valeur à lire.

Il copie ensuite ces valeurs dans l’espace utilisateur.

Pour éviter que la commande cat n’affiche rien, dans un premier temps la fonction renvoie le nombre max de bit que le buffer peut contenir. A noter que cette valeur est aléatoire. Au deuxième appelle de la fonction par cat la fonction renvoie directement 0.

### Write

La fonction write lit la chaine de caractères écris avec la commande echo. Si le nombre de caractères est supérieur à 9 (8 bits pour les LED plus le retour à la ligne), la fonction met directement toutes les LED à 0. Si le nombre de caractère est inférieur ou égal, la fonction vérifie pour chaque caractère s’il est égal à 1 ou 0 (sinon il est ignoré) et inscrit ce nombre dans une variable char. La fonction commence par le bit le plus à gauche et complète par des zéros à droite si la chaine est trop petite.

Si on inscrit « 1110 » on obtient alors l’octet « 11100000 ».

La plus grande difficulté de la partie driver fut l’installation du drivers sur la machine virtuel.