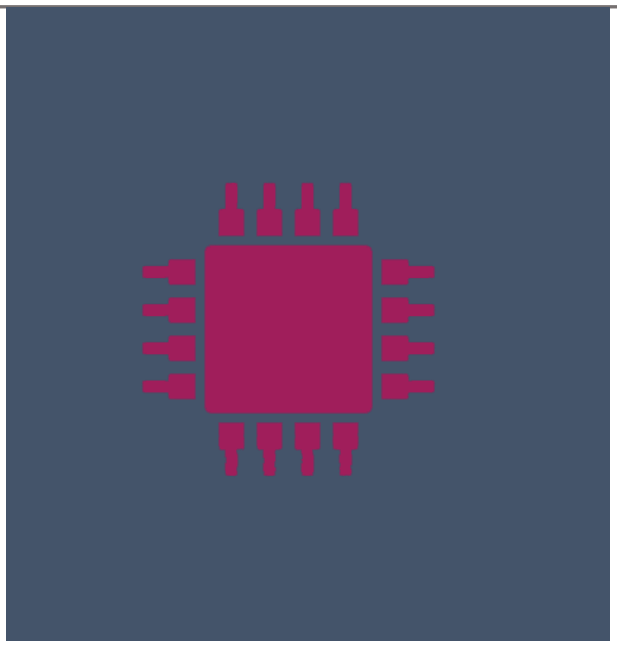

# Compte-Rendu Tp1-Chenillard

Microcontrôleur

Marion Escouteloup – Damien Dubois

CAMPUS
D'ENSEIGNEMENT SUPÉRIEUR
ET DE FORMATION PROFESSIONNELLE

# TABLE DES MATIERES

TP1-CHENILLARD

# I. Introduction

## 1. Présentation du sujet

Dans le cadre des travaux pratiques de microcontrôleurs, nous devons mettre en place un chenillard de LEDS. Ce chenillard a pour but d'aller successivement, une à une, 8 LEDS, avec un sens et une fréquence de clignotement variable. Pour contrôler ce délais, l'utilisateur doit activer une bouton poussoir (BP1) afin de faire varier ce temps. Il doit aussi pouvoir modifier de la même manière, sur un second bouton (BP2), le sens du chenillard. L'utilisateur devra également pouvoir choisir d'arrêter le chenillard pour recopier l'état de quatre switch sur quatre LEDS, et ceci par l'appui sur les deux boutons cités précédemment.

Le Microcontrôleur utilisé est un PIC 18.

## 2. Préparation

Pour débuter le projet, il nous est demandé d'importer un fichier ***config.c***, à inclure dans le projet, qui a pour but de configurer des registres du microcontrôleur pour son utilisation. Nous devons également créer un fichier .c ainsi que son .h **(Header),** pour créer les fonctions utilisé dans le fichier *main* du projet. Nous devons dans ces fichiers définir et programmer les différentes fonctionnalités utiles, tel que la fonction *INIT()* qui va initialiser le microcontrôleur pour notre application chenillard.

Tout d'abord nous choisissons de définir des mots clés relier aux PIN utilisées, ceci pour chaque LEDS, chaque boutons et chaque SWITCHS utilisés. Par exemple « Led_0 » pour la première LED. ce qui remplace *« LATDbits.LATD0 »* pour sa commande. En effet cela permet un code plus lisible.

# II. Les fonctions

## 1. INIT()

La fonction **INIT** va permettre de paramétrer le microcontrôleur au tout début de l'application. En effet, il nous faut définir les ports utilisés en entrées ou en sorties. Par déffinitions, nous initions :

- Les LEDS en **sorties**.
- Les boutons et SWITCHS en **entrées**.

Pour cela nous choisissons d'affecter ces entrés/sorties une par une, et seulement celles utilisées. Car souvent la programmation d'un microcontrôleur se fait en équipe et il ne faut pas modifier les paramètres déjà définis. Cela sera fait en utilisant par exemple « TRISGbits.TRISG3 = 1 » pour mettre la pin 3 du port G en entrée sans modifier le reste du port.

Pour notre application nous décidons d'utiliser des variables globales pour que toutes les fonctions puissent y avoir accès. Les valeurs par défauts sont initialisées dans la fonction *INIT()*.

## 2. RE_INIT()

La fonction **RE_INIT** permet de réinitialiser le chenillard dans le cas où l'utilisateur choisirait le mode recopie des SWITCHS sur les LEDS. En effet, le chenillard dépend de l'état précédent des LEDS, donc s'il y a deux LEDS allumées, le chenillard est faussé. Nous avons choisi qu'à la sortie du mode recopie, nous réinitialisons le chenillard.

## 3. T_SENS()

La fonction **T_sens()** prend en compte l'appui sur le bouton utilisateur qui permet de changer le sens du chenillard. En fonction de l'état précédent de la variable *sens*, le sens est inversé à l'appui sur le bouton. Le sens est initialisé au début du programme dans la fonction I**NIT()** et dans la fonction **RE_INIT()** à la sortie de la recopie des LEDS (fonction **T_switchs()**)

## 4. T_SWITCHS()

La fonction **T_switchs** recopie l'état des quatre interrupteurs switch sur les quatre LEDS correspondantes. Cette fonctionnalités s'active lors de l'appui simultannément sur les deux boutons poussoirs, et s'exécute au relâchement d'un ou des deux boutons. Avant de sortir de la fonction, on appelle la fonction **RE_INIT** pour remettre le chenillard en état de fonctionnement.

## 5. T_VITESSE()

La fonction **T_vitesse** prend en compte la vitesse actuel. Elle applique aussi, lors de chaque sur le bouton poussoir correspondant, un changement de la variable vitesse. À l'état initial on est en vitesse 1. Un appui fait passer à la vitesse suivante jusqu'à la vitesse 4 puis on repart à la vitesse 1. Cette variable vitesse permettra de choisir la temporisation effectuée dans la fonction **T_tempo()**. Cette fonctionnalité alterne entre quatre temps différents, comme expliqué dans la fonction **T_tempo()** ci-dessous.

## 6. T_TEMPO()

La fonction **T_tempo** met en place une temporisation pour cadencer le chenillard. En fonction de la variable seconde, la temporisation sera de 0.5 secondes en vitesse 1, puis doublée à la vitesse suivante, soit :

- Vitesse 1 : 0.5 s
- Vitesse 2 : 1 s
- Vitesse 3 : 2 s
- Vitesse 4 : 4 s

Pour la temporisation nous utilisons la fonction **__delay_ms(x)** où x doit être inférieur à 75 *(cf. Librairies générales langages C).* Cette fonction applique un délais en millisecondes. Au vu de notre contrainte qui est de pouvoir placé des délais supérieurs à 75ms, nous avons une boucle for permettant de rajouter un facteur 100 à nos délais. Par exemple, pour une temporisation de 0.5s, nous réalisons 100 fois un délais de 5ms.

## 7. T_CHENILLARD()

La fonction chenillard a pour but de réaliser l'exécution du décalage entre deux LEDS. Pour ce faire, le programme récupère le sens de marche du chenillard, pour décaler les LEDS de gauche à droite (ou inversement en fonction du sens). Il y a huit LEDS, arriver à la fin du chenillard, on repart à la LEDS du début.

# III. Main

Le main est une fichier .c qui permet d'exécuter le programme complet. Ici, nous appelons à tour de rôles les différentes fonctions. Dans l'ordre :

- **ETAPE 0 :** initialisation et configuration des ports sur le microcontrôleur

Nous rentrons à présent dans une boucle *while(1)*, nous permettant de faire tourner notre programme à l'infini.

- **ETAPE 1 :** applications et mise en états des LEDS 4 à 7 en fonction des SWITCHS 1 à 4. A condition que les deux boutons soient appuiés
- **ETAPE 2 :** récupération des données utilisateurs pour l'application du sens et du tempo du chenillard
- **ETAPE 3 :** Décalage de la LED en fonction des éléments récupérés précédemment et affichage de LEDS
- **ETAPE 4 :** Mise en place du délais en fonction du tempo sélectionné en E2 avant de repartir en E1

Pour le bon fonctionnement de l'ensemble des appels de fonction, et afin de limiter le nombre de paramètres d'entrées et de sorties de chaque fonction, les variables *sens, vitesse et Leds* sont initialisées dans le fichier main.c (cf. 3. INIT(), page 3).

Afin que tout fonctionne bien, il est important d'inclure la librairie préalablement développée pour accueillir les différentes fonctions. Cette librairie se nomme dans notre cas *"FISA26Biblio.h"*.

# IV. Conclusion

Pour conclure ce compte rendu, nous avons pu tester les fonctionnalités de notre code qui était opérationnel sur la mallette d'essai mise à notre disposition.

Utilisation mémoire :

```
Memory Summary:
    Program space        used   21Eh (   542) of 20000h bytes   (  0.4%)
    Data space           used     5h (     5) of   F16h bytes   (  0.1%)
    Configuration bits   used     7h (     7) of     7h words   (100.0%)
    EEPROM space         used     0h (     0) of   400h bytes   (  0.0%)
    ID Location space    used     8h (     8) of     8h nibbles (100.0%)
```