

Objectifs

- Identifier les principaux éléments de la carte
- Créer un projet avec Mplab
- Compiler et transférer un fichier dans le micro
- Utiliser les leds et les boutons

Introduction

L'objectif de ce module est de programmer une GameGirl: il s'agit d'une carte électronique comportant un écran LCD, un joystick, 4 boutons, un microcontrôleur et une liaison série. Durant les 5 premières séances, vous écrirez les bibliothèques permettant de configurer le microcontrôleur et d'utiliser les différents périphériques :

- GameGirl.h: librairie qui contient la configuration des entrées/sorties et l'initialisation des registres du microcontrôleur.
- LCD_128x64.h : librairie qui contient la gestion de l'afficheur graphique 128 x 64 pixels.
- Joystick.h : gestion du joystick.

Au cours de 5 dernières séances, vous aurez à programmer l'application de votre choix en utilisant les bibliothèques citées ci-dessus.

Evaluation

Au cours de la dernière séance, votre projet sera évalué selon 3 critères :

- La structure et la clarté de votre projet
- Les commentaires dans votre code
- L'état d'avancement de votre application

Création du projet

Lancer Mplab et créez un nouveau projet : **Project > Project wizard**

Sélectionner le microcontrôleur **PIC18F452**.

Choisir le compilateur C18 : **Microchip C18 Toolsuite**

Nommer votre projet et enregistrer le dans le répertoire qui vous est réservé¹.

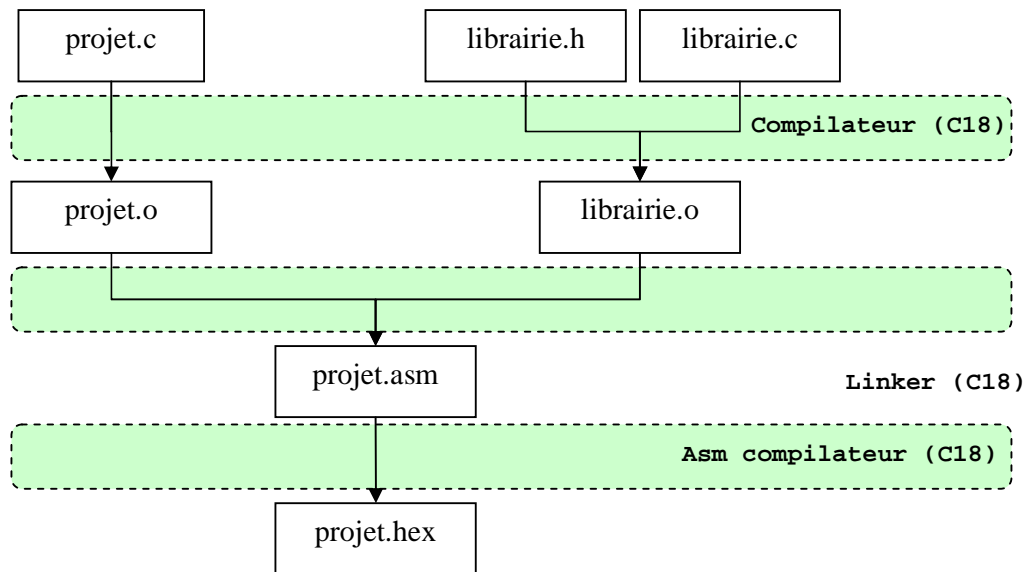
Ajouter le linker au projet : **C:\MCC18\lkr\18f452.lkr**

Ajouter également le fichier de gestion du PIC18F452 : **C:\MCC18\h\p18f452.h**

Créer un nouveau fichier main.c et ajouter le au projet.

¹ Demandez à l'enseignant

Voici la façon dont le programme final (en langage machine) va être généré :



Chaque fichier C est compilé indépendamment. Dans l'exemple ci-dessus, les librairies et le programme principal sont compilés séparément. Ensuite, tous les fichiers .o générés sont liés pour former un unique fichier assembleur avant sa compilation finale en langage machine.

Ajouter dans ce fichier le code suivant et vérifier que la compilation se passe sans problème.

```

#pragma config WDT = OFF      // Pas de watchdog
#pragma config OSC = HS       // HS pour un quartz 20MHz

void main(void){}
  
```

Les deux premières lignes sont des directives de programmation, elles servent à désactiver le chien de garde (WDT = OFF) et à préciser que microcontrôleur va fonctionner avec un quartz à haute vitesse (OSC = HS).

Ne continuer pas tant que la compilation ne s'est pas parfaitement déroulée.

Librairie GameGirl

Créer deux nouveaux fichiers **GameGirl.h** et **GameGirl.c** et ajouter-les au projet.
Utilisez les lignes suivantes dans le fichier **GameGirl.h** qui permet de définir la macro **__GAMEGIRL__** afin d'éviter plusieurs appels à la fonction.

```
#ifndef __GAMEGIRL__
#define __GAMEGIRL__

// Définitions des constantes et des prototypes des fonctions
...

#endif
```

Avec le compilateur C18, il est possible d'adresser les registres bit à bit. Par exemple pour le bit 2 du port D, il est possible d'utiliser :

PORTDbits.RD2

L'ensemble des registres est déclaré dans le fichier **p18f452.h**. Vous devrez fréquemment vous y référer durant le projet.

Dans le fichier **GameGirl.h**, déclarer les entrées / sorties du microcontrôleur relatives aux boutons et aux leds en vous inspirant de la syntaxe suivante :

```
#define GreenLed PORTEbits.RE0    // Led verte
                                   // !! Active à l'état bas !!
```

Pour les définitions, utiliser les noms suivants :

Bouton poussoir vert	Button_Green
Bouton poussoir bleu	Button_Blue
Bouton poussoir rouge	Button_Red
Bouton poussoir jaune	Button_Yellow
Led rouge	Led_Red
Led verte	Led_Green

Penser à commentez au fur et à mesure l'écriture de votre projet.

Ajouter une fonction **void ConfigProcessor(void)** à la librairie.

La configuration des ports en entrée ou en sortie se fait par l'intermédiaire des registres TRISx (pour TRI State). Par exemple pour le port A, utilisez l'instruction suivante pour configurer un bit sur deux en entrée (1 → I → Input et 0 → O → Output).

```
TRISA = 0b10101010; // Button | RedLed | XXXXXX | ...
```

Prise en main de la carte et des logiciels

Configurer les ports reliés aux leds et aux boutons poussoirs selon qu'ils doivent être en entrée ou en sortie. Configurer toutes les autres broches en entrée.

Le microcontrôleur est équipé de convertisseurs analogique/numérique. Dans un premier temps, utilisez l'instruction suivante pour forcer la configuration numérique des ports. Nous verrons plus tard (dans Joystick.h) les détails de cette fonction.

```
OpenADC ( ADC_FOSC_2 &  
          ADC_RIGHT_JUST &  
          ADC_OANA_0REF ,  
          ADC_INT_OFF );
```

Utiliser le guide des librairies situé dans **C:\MCC18\doc** pour chercher à quelle librairie appartient la fonction **OpenADC**.

Tests

Ajouter un appel à la fonction **ConfigProcessor** depuis le programme principal.

Gardez toujours à l'esprit qu'un microcontrôleur n'a pas de système d'exploitation. Lorsqu'une application est terminée, elle n'est pas tuée par l'OS. Elle continue toujours à s'exécuter en mémoire. Pour terminer un programme ajouter une boucle infinie.

```
while (1);
```

Ecrire le main de façon à ce que

- la led rouge s'allume lors d'un appui sur le bouton rouge,
- la led jaune clignote lors d'un appui sur le bouton jaune
- la led verte s'allume lors d'un appui sur le bouton vert,
- la led bleue clignote lors d'un appui sur le bouton bleu

Vous trouverez dans le guide des librairies les détails des fonctions nécessaires (par exemple les temporisations)

Transférez votre projet dans le microcontrôleur selon les instructions qui vous seront données durant le TP.