Jeux de lumières

Découverte de la programmation sur microcontrôleur

BARBESANGE Benjamin, GARCON Benoît

avril 2014

Table des matières

Remerciements

Nous tenons à remercier dans un premier temps M. Cheminat, responsable du projet qui nous a encadrés tout au long de celui-ci.

Nous tenons également à remercier les membres du club robotique de l'ISIMA (Isibot) et particulièrement François, président du club, ainsi que Noël pour l'aide qu'ils nous ont fournie.

Introduction

Présentation du projet

Dans le cadre de la deuxième année de Prep' ISIMA, il est demandé d'effectuer un projet de groupe occupant 24h de travail. Le sujet que nous avons ainsi choisi est le sujet "Jeux de lumières".

Afin de pouvoir réaliser ce projet, le matériel suivant a été mis à notre disposition :

- Une carte EasyPIC
- Un microcontrôleur (réf. P18F452)
- 6 résistances de 1 kΩ
- Une LED RGB
- Une plaquette de branchement ainsi que des fils de connexions

Microcontrôleur PIC

La partie la plus importante du matériel est le microcontrôleur. Il va nous permettre d'interagir avec les différents composants grâce à un programme qu'il faut préalablement charger. Le microcontrôleur utilisé dans ce projet a pour référence P18F452.

Il possède 40 pins. Les pins RA, RB, RC, RD et RE (au nombre de 8 pour les premières et 4 pour les RE) sont utilisables comme entrées/sorties standards. Toutefois, les pins RC1 et RC2 sont aussi capable de générer un signal PWM.

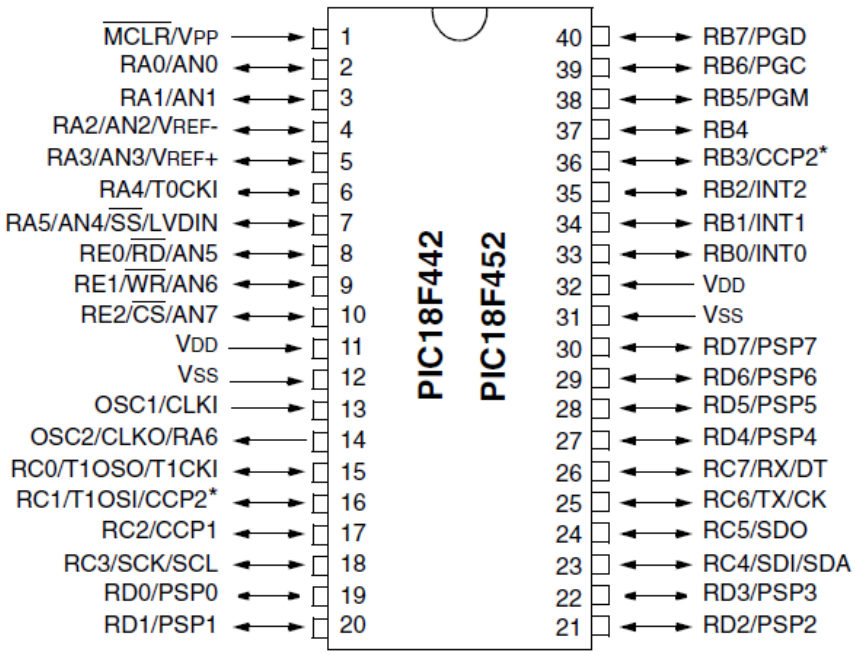

Figure 1 - Microcontrôleur P18F452

Carte EasyPIC

La carte EasyPIC fournie est le modèle 4. C'est la base du projet, qui fait le lien entre les différents composants. La carte est composée d'une matrice de LED dite de "contrôle", ainsi que d'un écran GLCD sur lequel il est possible d'afficher des images. Ce sont les deux principaux éléments sur lesquels nous intervenons.

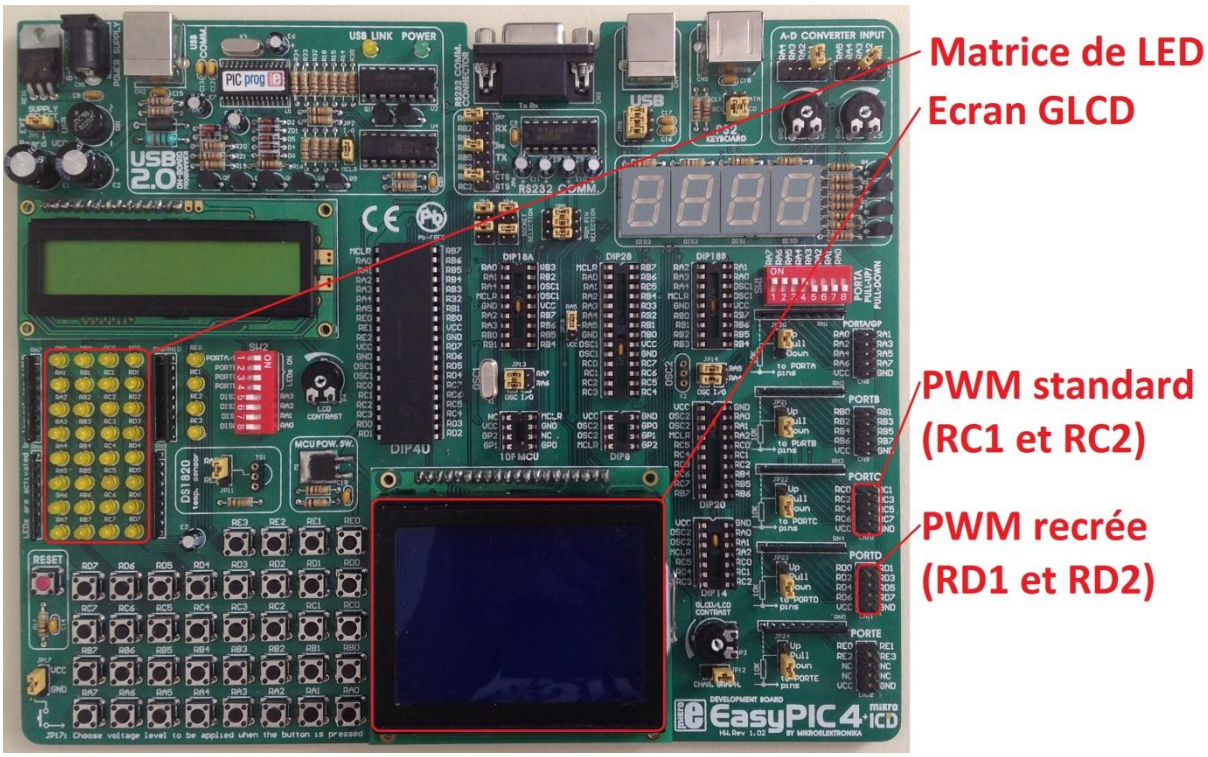

Figure 2 - Carte EasyPic 4 et ses composants

LED RGB

En plus de la carte PIC, une LED RGB est mise à disposition. Celle-ci vient accompagnée de plusieurs résistances de 1 kΩ (afin d'éviter de griller la LED). De plus, une plaquette de branchement ainsi que des fils viennent s'ajouter afin de pouvoir relier cette LED à la carte PIC, qui va ainsi déterminer de fonctionnement de cette LED.

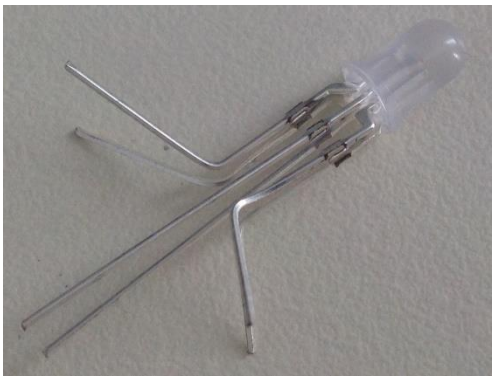

Figure 3 - LED RGB

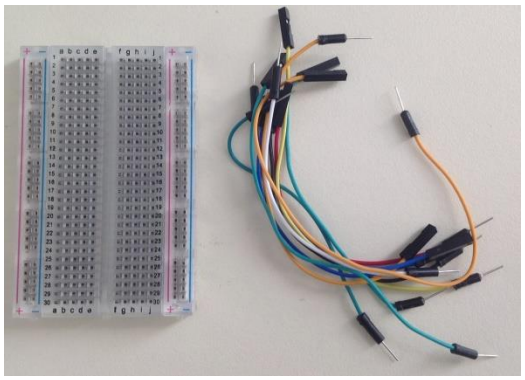

Figure 4 - Plaquette de branchement et fils

Présentation du programme

Plusieurs langages de programmation sont disponibles pour le microcontrôleur. Etant plus à l'aise avec le langage C, il est donc naturel que celui-ci ait été utilisé.

L'intégralité du code est consultable sur github à l'adresse suivante : https://github.com/BBS007/Jeux_lumiere.

Afin d'améliorer la lisibilité du code, celui-ci est commenté et fractionné en plusieurs fichiers, chacun regroupant les fonctions propres à la manipulation d'un composant.

Fichier main.c

Ce fichier est le cœur du programme. Il contient la boucle principale servant à appeler les différentes fonctions de manipulation des composants. Il est possible de lancer les différentes fonctions du programme en appuyant sur les boutons de la carte.

Voici la liste des fonctions associées à leurs boutons :

- RB0 : Affiche le logo de l'ISIMA sur l'écran GLCD
- RB1 : Affiche un décompte de 9 à 0 sur la matrice de LED
- RB2 : Fait varier l'intensité de la LED RGB sur le port PWM natif
- RB3 : Fait varier l'intensité de la LED RGB sur le port PWM recrée

Il est recommandé d'appuyer sur le bouton reset entre différents appels de fonction.

Fichier chiffre.c

Dans ce fichier se trouvent les fonctions permettant les affichages de chiffre sur la matrice de LED.

Fichier glcd.c

Ce fichier répertorie les fonctions propres à l'affichage du logo de l'ISIMA sur l'écran GLCD.

Fichier ledRGB.c

Ce fichier permet de manipuler la LED RGB avec le port PWM natif et celui recréé.

Partie 1 - Découverte du matériel

Le projet a débuté le Mardi 21 Janvier. Pour assurer la compatibilité des machines utilisées avec la carte, il a fallu installer les drivers ainsi que les programmes associés :

- mikroProg Suite For PIC (interface avec la carte)
- mikroC PRO for PIC (l'Environnement de Développement Intégré).

Afin de vérifier que la carte ainsi que les composants fonctionnaient correctement, il a fallu charger le programme LedBlinking qui est un programme de test et permet de faire clignoter chaque élément sur la carte. Avant de charger ce programme, nous avons préalablement changé le modèle du microcontrôleur (P18F452) ainsi que l'horloge dans l'IDE (20 Mhz).

Lors de l'alimentation de la carte, les voyants POWER et USB se sont allumés, et une faible lueur a jaillit d'une des LED de la matrice. Les éléments de la carte ne clignotant pas comme ils devaient le faire (grâce au programme LedBlinking), nous avons donc essayé de trouver la source du problème.

Pour cela, nous avons d'abord revérifié si les paramètres de l'IDE étaient en accord avec ceux de la carte. Nous avons ensuite réinstallé les drivers ainsi que les programmes associés avec la carte, pour enfin changer de système d'exploitation.

Nous nous sommes donc rendu dans le local d'ISIBOT et les conclusions ont été rapides : le EasyPIC 4 était défectueux, nous en avons donc reçu une nouvelle ainsi qu'un nouveau quartz de 8Mhz par la même occasion.

Le fonctionnement du matériel étant maintenant assuré, nous avons pu commencer à travailler sur le projet.

Partie 2 - La matrice de LED

La carte Pic dispose d'une matrice de LED de contrôle qu'il est possible d'alimenter. Il nous est demandé de réaliser un décompte sur cette matrice de LED. Etant donné que la taille de la matrice est 8 × 4 nous nous sommes résolus à réaliser un décompte de 9 à 0.

Il a fallu dans un premier temps établir un alphabet numérique de 8 × 4 "cases". Une fois celui-ci établit, nous avons pu découvrir le fonctionnement de ces LED qu'il faut simplement d'alimenter dans le programme. Il suffit d'indiquer un nombre binaire sur 8bits représentant une colonne de LED, dans lequel chaque bit représente une LED de cette colonne. Un bit est mis à '1' pour que la LED soit dans l'état allumé ou à '0' pour que la LED soit dans l'état éteint. Ceci sous condition que le cavalier 17 (en bas à gauche de la carte) soit à la bonne position, sur VCC en non pas sur la terre.

Nous nous sommes rapidement rendu compte que les LED RA4, RA6 et RA7 ne s'allumaient pas. Après une vérification dans la documentation du PIC, il s'est avéré que les LED RA6 et RA7 sont utilisées pour l'oscillateur et non pour les entrées/sorties.

Après une visite au club robotique, il nous a été demandé d'ignorer le non fonctionnement de celles-ci et de les considérer comme si elles s'allumaient. Nous avons cependant cherché un moyen de pouvoir allumer ces LED sans succès. En effet une des principales pistes étaient de se servir d'un autre oscillateur mais cette piste n'a pas été fructueuse.

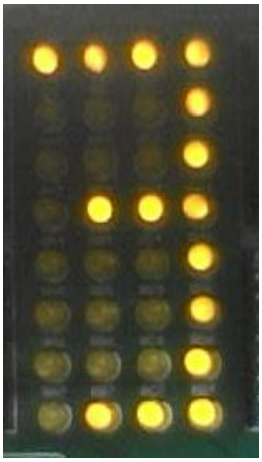

Figure 5 - Affichage du chiffre 3 sur la matrice de LED

Partie 3 - L'écran GLCD

La carte Pic fournie disposant d'un écran GLCD, nous avons trouvé qu'il pouvait être intéressant d'exploiter ce composant, même si celui-ci n'était pas à considérer dans le projet dans un premier temps.

Pour utiliser ce composant, nous avons décidé de faire afficher le logo de L'ISIMA. Nous avons donc téléchargé celui-ci puis grâce à un logiciel de traitement d'image (Gimp) nous avons transformé l'image en image bicolore. Puis grâce à un programme de la suite officielle de mikroelectronika nous avons pu transformer la bitmap en un tableau d'octets constants (const char[1024]).

Figure 6 - Affichage du logo ISIMA sur l'écran GLCD

Il a ensuite été simple d'afficher le logo à l'aide de la bibliothèque adaptée. Pour aller plus loin nous avons voulu regarder plus précisément le fonctionnement de l'écran. Tout d'abord il est composé de 64 lignes de 128 pixels. Ces 8192 pixels sont divisés en deux côtés de dimensions 64 × 64 pixels. Chacun de ses côtés est divisé en 8 pages de dimension 8 × 64. Enfin toutes ces pages sont composées de 64 octets qui décrivent l'état des 64 colonnes de 8 pixels.

Ainsi nous avons compris le sens de notre tableau d'octet dont chaque valeur décrit donc une "colonne" de 8 pixels. Nous avons ensuite écrit une fonction qui permet d'afficher à l'écran une structure un peu plus intuitive (ligne par ligne).

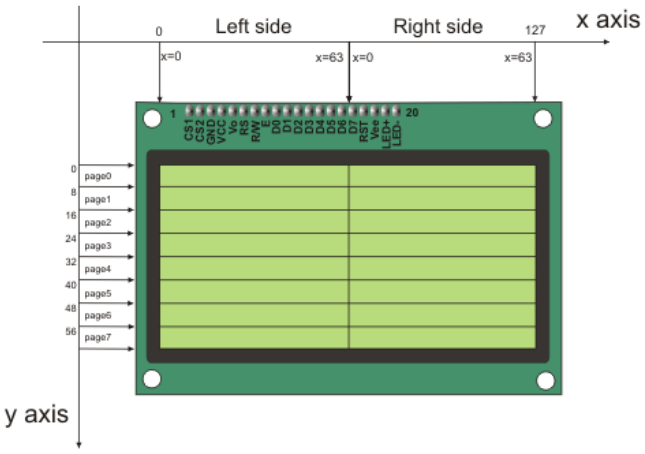

Figure 7 - Fonctionnement du GLCD

Partie 4 - La LED RGB

Nous disposons à présent d'une LED RGB à quatre pattes, de six résistances identiques de 1 kΩ et de deux LED dites classiques.

Port PWM "classique"

Les deux ports PWM (Pulse With Modulation) correspondent aux ports RC1 et RC2. Ces ports permettent de faire varier l'intensité du courant transmis, notamment en modifiant le rapport cyclique, ce qui a pour effet de créer des intensités de lumières plus ou moins importantes sur une LED.

Dans un premier temps, afin de tester le fonctionnement du port PWM, nous avons utilisé les LED classiques que nous avons branchées en série.

Après avoir reçu la LED RGB nous l'avons branchée en série avec les résistances associées nous avons réussi à faire clignoter chaque couleur allant de l'intensité la plus faible à l'intensité la plus forte.

N'ayant que deux ports PWM nous ne pouvons donc utiliser que deux couleurs à la fois. Nous avons choisi dans un premier temps d'utiliser la LED rouge et la LED bleue afin de mieux les distinguer.

Afin d'utiliser le maximum des possibilités de la LED RGB nous avons décidé de faire un fondu de la couleur bleue à la couleur rouge puis inversement en passant par toutes les gammes d'intensité disponibles. La prise en main de la librairie PWM a été assez rapide. En effet une fonction très simple permet de régler le rapport cyclique : paramétrable de 0 à 255 cela règle la largeur d'impulsion de RC1 et RC2 respectivement de 0 à 100%. Il a alors été assez aisé d'établir des mélanges de couleurs.

Création d'un port PWM

Une fois notre travail terminé avec les ports PWM intégrés sur la carte, il nous a été demandé de recréer un tel port sur une patte n'ayant pas la possibilité de créer de base un signal de type "carré". Cela nous permettra d'utiliser les 4 composantes de la LED RGB.

Dans un premier temps, nous avons effectué des mesures sur le port PWM normal afin de déterminer la demi-période à recréer. La fréquence de celui-ci est réellement de 5kHz comme nous l'avions paramétré dans le code.

Ensuite nous avons commencé à programmer un équivalent à la fonction PWM_Set_Duty() pour l'ensemble des pattes RD. Algorithmiquement très simple, cette fonction s'est avérée plus complexe. Nous avons dû simplifier notre boucle for car son itérateur provoquait des variations de fréquences, passer de delay() à delay_ms() à delay_us() pour augmenter la fréquence de notre signal.

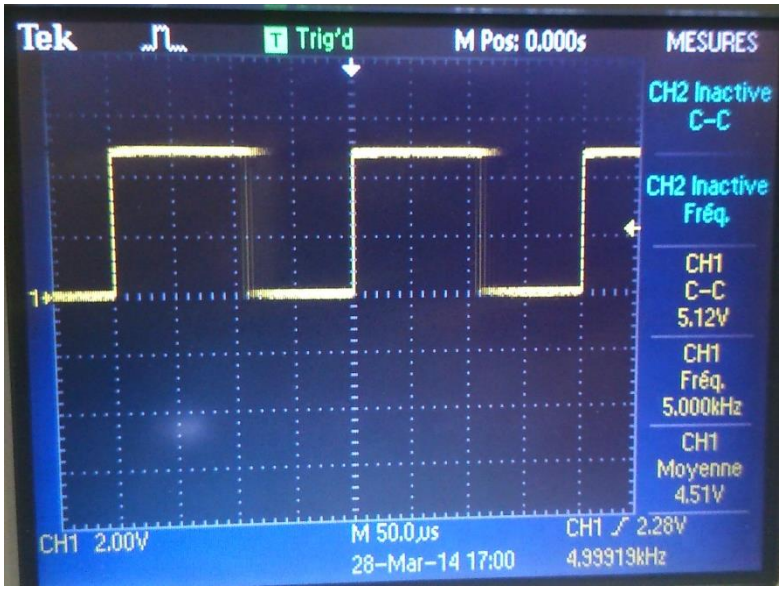

Figure 8 - Mesure d'un vrai port PWM

Après avoir achevé la programmation des ports "pseudo" PWM, les mesures ont montré que nous avions réussi à recréer un signal carré de fréquence moyenne d'environ 640Hz et de rapport cyclique paramétrable de 0 à 100% avec un pas de 0,5%.

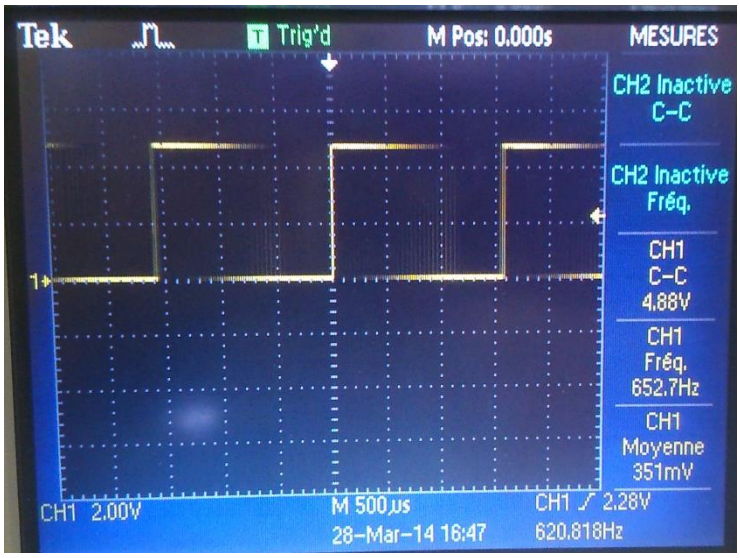

Figure 9 - Mesure d'un pseudo port PWM

Le temps de calcul induit par notre code est trop grand pour atteindre une fréquence de 5kHz, une solution aurait été d'utiliser directement les registres qui servent justement à implémenter les fonctions delay pour créer les nôtres. Toutefois à l'œil nu la différence entre notre PWM et les vrais PWM est imperceptible.

Partie 5 - Essais divers

Texte sur écran LCD

Afin d'étendre nos capacités sur la carte PIC, nous avons tenté de manipuler l'écran LCD. Après avoir chargé un programme test affichant la marque de la carte, nous avons voulu modifier le texte de ce programme exemple afin de faire afficher notre propre texte.

Cependant après chargement du nouveau programme sur la carte, il s'est avéré que le texte ne s'affichait pas sur l'écran. Après de multiples recherches dans la documentation des fonctions ainsi que sur internet, nous n'avons pas réussi à résoudre ce problème d'affichage et avons décidé de nous réorienter vers le projet initial en découvrant le port PWM.

Jeu de la vie sur GLCD

Suite à l'affichage du logo de l'ISIMA sur le GLCD, nous avons essayé de pousser l'expérience jusqu'à la conception d'un automate cellulaire type jeu de la vie. L'idée originale était de se servir du logo de l'ISIMA comme graine pour l'automate.

Le développement de cette idée a nécessité de pouvoir afficher plus simplement des images sur le GLCD, d'où l'implémentation d'une fonction permettant d'afficher un tableau à deux dimensions. L'algorithme de l'automate n'était pas un problème non plus, ce qui nous a le plus gêné reste vraiment les problèmes de mémoire rencontrés.

En effet, à ce moment nous avons vraiment trouvé une des premières limites de l'embarqué : la mémoire. Dans l'algorithme de notre automate nous avions besoin d'un tableau contenant l'état au temps $t-1$ et d'un autre avec l'état au temps t. Malheureusement nous n'avons pas pu déclarer un tableau autre que constant puisque à chaque fois le compilateur nous rappelait à l'ordre, et lorsque nous avons cru pouvoir nous affranchir de ce problème, la mémoire du matériel était insuffisante.

Ce fut donc l'occasion de nous rendre vraiment compte de la différence des contraintes entre de la programmation traditionnelle et de l'orientée embarqué.

Bilan

Au terme de ce projet nous sommes maintenant quelque peu familiers de la programmation sur Pic. Ce fut pour nous deux notre première expérience sur de la programmation orientée embarqué. Démarrant confiants sur l'aboutissant du projet nous avons dû à plusieurs reprises remettre en cause notre façon de faire et de penser pour pouvoir aboutir à nos résultats actuels.

Il a été intéressant de voir une autre façon de programmer : les contraintes étaient différentes, il fallait s'intéresser beaucoup plus au matériel, etc. Lorsqu'il y avait une erreur le doute planait toujours pour savoir si elle venait du matériel ou du code. Malgré les difficultés du début nous avons appris à manipuler quelques subtilités de la carte et nous avons apprécié conduire à bien ce projet.

Table des illustrations