

# MyLab2 GB

## Emulateur de Gameboy sur MyLab2

Orphée Antoniadis

Projet de semestre - Prof. Fabien Vannel

Hepia ITI 3ème année

Semestre d'automne 2017-2018

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Objectif . . . . .	3
1.2	Méthode . . . . .	3
1.3	Schéma . . . . .	3
<b>2</b>	<b>MyLab2</b>	<b>3</b>
2.1	Introduction . . . . .	3
2.2	Périphériques utilisés . . . . .	3
2.2.1	Boutons . . . . .	3
2.2.2	Joystick . . . . .	4
2.2.3	Ecran LCD . . . . .	4
2.2.4	Dalle tactile . . . . .	4
<b>3</b>	<b>Gameboy</b>	<b>5</b>
3.1	Introduction . . . . .	5
3.2	CPU . . . . .	6
3.2.1	Architecture . . . . .	6
3.2.2	Registres . . . . .	6
3.2.3	Instructions . . . . .	6
3.3	Mémoire . . . . .	6
3.3.1	Memory map . . . . .	6
3.3.2	Cartouche . . . . .	6
3.3.3	Bootstrap . . . . .	6
3.3.4	DMA . . . . .	6
3.4	Interruptions . . . . .	6
3.4.1	Activer une interruption . . . . .	6
3.4.2	Demander une interruption . . . . .	6
3.4.3	Traiter une interruption . . . . .	6
3.4.4	Comportement de HALT . . . . .	6
3.5	Timer . . . . .	6
3.5.1	Timer controller . . . . .	6
3.5.2	Divider register . . . . .	6
3.6	GPU . . . . .	6
3.6.1	Contrôleur LCD . . . . .	6
3.6.2	Affichage . . . . .	6
3.7	Entrées . . . . .	6
<b>4</b>	<b>Émulateur</b>	<b>6</b>
<b>5</b>	<b>Conclusion</b>	<b>6</b>

# 1 Introduction

## 1.1 Objectif

L'objectif principal de ce projet était de développer un émulateur de la console de jeux vidéos de Nintendo appelée Gameboy pour la carte de d'extension MyLab2 développée à l'hepia. Un emulateur est un logiciel permettant d'imiter le comportement physique d'un matériel informatique. Le but ici est donc d'imiter le comportement du processeur de la Gameboy avec tous ces périphériques.

Il a fallut dans un premier temps comprendre le fonctionnement de la Gameboy. Le projet a donc commencé par un travail de recherche et de documentation. Etant donné que Nintendo n'a jamais rendu public la datasheet de sa console, toutes les informations récoltées ont été obtenues après un travail de reverse engineering effectuée par la communauté. Heureusement, la communauté de développeurs d'émulateurs est très active, de nombreux forums et blogs existent ce qui a simplifié la recherche d'informations. Toutes les sources que j'ai trouvé seront listées à la fin de se document et pourront faire office de base de donnée complète pour quiconque voulant en apprendre plus sur la console ou bien même voulant développer son propre émulateur.

## 1.2 Méthode

## 1.3 Schéma

# 2 MyLab2

## 2.1 Introduction

La MyLab2 est une carte d'extension pour les kits de développement des microcontrôleurs LPC1769 et LPC4337 de NXP. Cette carte a été développée à hepia, au laboratoire de systèmes numériques. Elle vient se superposer aux cartes de NXP via des connecteurs broches.

Le LPC1769 de NXP est un microcontrôleur ARM Cortex-M3 ayant une fréquence d'horloge pouvant aller jusqu'à 120MHz. Ce dernier propose une mémoire flash de 512kB (mais seulement la moitié est programmable avec la version gratuite de LPCXpresso) ainsi qu'une SRAM de 64kB. A noter que les 64kB de SRAM sont discontinus et qu'il y a en réalité deux banques de RAM de 32kB chacune. Le LPC4337 de NXP est un microcontrôleur ARM Cortex-M4 ayant une fréquence d'horloge pouvant aller jusqu'à 204MHz mais inclut aussi un coprocesseur ARM Cortex-M0 pouvant aller à la même fréquence d'horloge. La SRAM de ce dernier fait 136kB.

Pour ce projet, le microcontrôleur utilisé sera le LPC1769 pour un soucis pratique. En effet, nous avons déjà eu à utiliser cette carte pour le cours de microcontrôleurs et périphériques de deuxième année ce qui fait que j'avais déjà une grande partie de la librairie de gestion des périphériques de la carte d'extension prête. C'était un gain de temps non négligeable qui m'a permis de me concentrer sur l'émulation de la Gameboy.

Le projet pourrait tout de même être repris et porté sur le LPC4337 afin d'avoir de meilleures performances étant donné que sa fréquence d'horloge est deux fois plus grande, qu'il dispose de deux processeurs permettant une parallélisation des tâches et qu'il y a plus de RAM.

## 2.2 Périphériques utilisés

### 2.2.1 Boutons

Les boutons A et B de la MyLab2 sont de simples pins GPIO. Le LPC1769 propose des interruptions (EINT3) sur les ports 0 et 2 du GPIO. Ici, le bouton A est sur la pin 10 du port 2 (P2.10) et le bouton B est sur la pin 19 du port 0 (P0.19). L'interruption peut se faire sur le flanc montant ou descendant (ou les deux). Pour l'activer, il faut mettre le bit correspondant à 1 dans le registre `IntEnF` (flanc descendant) ou `IntEnR` (flanc montant). Lors d'une interruption, il faut regarder laquelle a eu lieu dans le registre `IntStatF/IntStatR` puis la quittance en mettant le bon bit à 1 dans le registre `IntClr`.

### 2.2.2 Joystick

Le joystick de la MyLab2 est aussi relié à une pin GPIO mais contrairement aux boutons A et B, il est relié au port 1 du GPIO. Il n'y a pas d'interruptions sur le port 1 du GPIO, il faut donc vérifier l'état des pins du joystick de manière régulière. Pour se faire, un timer sera utilisé qui provoquera une interruption (TIMER0) toutes les 10ms. La routine d'interruption appelle la fonction `joystick_handler` qui prend comme argument une fonction de callback (un pointeur sur une fonction qui sera appelée si une des directions du joystick est appuyée), l'argument de cette dernière, et le mode de vérification (POLLING ou TRIGGER). En mode POLLING, la fonction de callback sera appelée qu'une seule fois si le joystick est maintenu appuyé contrairement au mode TRIGGER où la fonction sera appelée tant que le joystick n'est pas relâché. La fonction de callback doit avoir comme prototype :

```
void joystick_callback(uint8_t pos, uint8_t edge, void *arg)
```

avec `pos` qui est la position du joystick, `edge` qui est le flanc (montant ou descendant) et `arg` qui contient l'argument de la fonction (peut être à `NULL`).

### 2.2.3 Ecran LCD

La MyLab2 utilise le bus de communication SPI pour communiquer avec l'écran. Afin d'initialiser le SPI, il faut d'abord sélectionner les bonnes pins dans les registres appropriés (PINSEL0 et PINSEL1). En effet, les pins sont configurées par défaut sur des pins GPIO. Pour le LPC1769 ce sont donc les bits 31 :30 de PINSEL0 et 5 :0 de PINSEL1 qu'il faut modifier.

Il faut ensuite fixer la valeur de la fréquence de transmission. Un registre de configuration du SPI le permet. C'est le registre SPCCR. La valeur donnée à ce registre va diviser la valeur de l'horloge du SPI. Pour atteindre une fréquence de transmission maximale, il faut mettre l'horloge du SPI à 100MHz en modifiant le registre PCLKSEL0 puis fixer la valeur du registre SPCCR à 10 pour avoir une fréquence de transmission de 10MHz ( $\frac{100}{10}$ ). Pour finir il faut juste activer la communication SPI en la mettant en master mode. Ceci se fait dans le registre SPCR (bit 5 à 1).

L'écriture et la lecture du bus SPI se fait à l'aide du même registre pour le LPC1769. Ce registre est le registre SPDR. Pour écrire, il faut fixer la valeur du registre aux données à envoyer puis attendre que le flag de confirmation d'envoi soit mis à 1. Ce flag est le bit 7 du registre SPSR.

Pour venir lire sur le bus SPI, il faut d'abord envoyer 0xFF (donc en utilisant la fonction d'écriture) puis venir lire sur ce même registre (buffer bi-directionnel).

L'écran LCD accepte 2 types de données, les instructions et les arguments d'instruction. La pin DC (pin GPIO 1.16) permet de contrôler ce qui va être envoyé. Le DC est à 0 lors de l'envoi d'une instruction et à 1 pour l'envoi d'un argument. La liste complète des instructions est disponible dans la documentation de l'écran LCD. Pour ce projet, seulement 3 de ces instructions sont utilisées. Celle pour sélectionner la colonne d'écriture en pixels (instruction 0x2A), celle pour sélectionner la ligne d'écriture en pixels (instruction 0x2B) et enfin, celle pour écrire dans la mémoire de l'écran (instruction 0x2C). La mémoire est l'ensemble des données qui composent l'intégralité des pixels de l'écran. Ici l'écran est de  $240 \times 320$  et un pixel est codé sur 16 bits donc la mémoire serait ces  $240 \times 320 \times 16$  bits. Pour allumer un pixel sur l'écran, il faut d'abord sélectionner la zone d'écriture en utilisant les instructions 0x2A et 0x2B, puis il faut écrire les données relatives à la couleur du pixel désirée en utilisant l'instruction 0x2C).

### 2.2.4 Dalle tactile

La MyLab2 communique avec la dalle tactile en utilisant le bus I<sup>2</sup>C. La détection d'appuie sur la dalle tactile se fait quand à elle par interruption GPIO. Il faut donc activer d'abord ces interruptions (comme pour les boutons A et B) puis récupérer les informations voulues dans la routine d'interruption. Les informations sont par exemple les coordonnées de position et peuvent être lues dans les registres de la dalle tactile dont les adresses sont disponibles dans la documentation. Pour ce projet, il n'y a eu besoin que de récupérer des coordonnées de la position détectée. Pour récupérer ces informations, il faut d'abord envoyer (donc écrire sur le bus I<sup>2</sup>C) l'adresse du registre à lire et ensuite venir lire dans ce registre.

A noter qu'il faut bien entendu initialiser le bus I<sup>2</sup>C ce qui est fait relativement de la même manière que le bus SPI. Petite différence juste pour fixer la fréquence de transmission qui se fait par deux registres, les registres I2SCLH et I2SCLL. Ces registres correspondent respectivement à la fréquence de l'horloge à l'état haut et à l'état bas. La fréquence du bus I<sup>2</sup>C en fast mode est de 400kHz et l'horloge périphérique est à 25MHz par défaut donc il faut mettre ces deux registres à 32 ( $\frac{25000}{32+32} = 400$ ). Il faut pour finir activer les interruptions I<sup>2</sup>C et créer l'algorithme représenté par un organigramme dans la documentation du LPC1769.

## 3 Gameboy

### 3.1 Introduction

La Gameboy est une console de jeux vidéos portable 8 bits développée et fabriquée par la firme japonaise Nintendo. La console a été mise en vente en 1989 et a connu un franc succès à travers le monde jusqu'à la fin de sa production en 2003. Nous allons dans cette partie étudier les caractéristiques et les périphériques de la console.

Le processeur de la Gameboy est le LR35902, processeur semblable au Zilog Z80 et cadencé à 4,19MHz. La mémoire de la console est de 64kB. Elle possède quatre boutons, A, B, START et SELECT, une croix directionnelle et une fente pour insérer les cartouches de jeux qui était située sur le haut, à l'arrière de la console. Elle est aussi dotée de hauts parleurs et d'un port série pour la communication entre deux consoles. Le son et la communication série ne seront pas emulés pour ce projet mais une amélioration future pourrait les rajouter. Divisons donc toutes ces parties en blocs.

## **3.2 CPU**

### **3.2.1 Architecture**

### **3.2.2 Registres**

### **3.2.3 Instructions**

## **3.3 Mémoire**

### **3.3.1 Memory map**

### **3.3.2 Cartouche**

### **3.3.3 Bootstrap**

### **3.3.4 DMA**

## **3.4 Interruptions**

### **3.4.1 Activer une interruption**

### **3.4.2 Demander une interruption**

### **3.4.3 Traiter une interruption**

### **3.4.4 Comportement de HALT**

## **3.5 Timer**

### **3.5.1 Timer controller**

### **3.5.2 Divider register**

## **3.6 GPU**

### **3.6.1 Contrôleur LCD**

### **3.6.2 Affichage**

## **3.7 Entrées**

## **4 Émulateur**

## **5 Conclusion**