Microprocesseur

TP4 : Liaison SPI

Cette séance permet de mettre en œuvre une liaison SPI entre le microcontrôleur et une mémoire externe (une mémoire Flash qui conserve ses données quand l'alimentation est coupée).

Les étapes de cette séance seront :
- Mise en œuvre assistée de l'écran LCD pour faciliter la validation logicielle
- Configuration de la liaison SPI.
- Lecture de la mémoire Flash externe.

1) Manipulation de l'écran LCD

Sur la partie supérieure de la carte trône fièrement l'écran LCD. Le contrôle de l'écran nécessite de gérer une commande parallèle manuellement (manipulation des bits directement par le processeur) et n'est pas standardisée. Nous n'allons donc pas passer de temps à en étudier le fonctionnement en détails. Deux fichiers sont fournis pour utiliser l'écran sans trop se poser de questions (`minimal_lcd.h` et `minimal_lcd.c`). Pour avoir réellement le temps de vous consacrer à la lisaison SPI, cette partie ne doit pas vous occuper plus de 30 minutes.

Le nombre réduit de fonctions doit permettre une prise en main rapide. Voici quelques détail sur son usage :

- **`void LCD_Init(unsigned char CursorOn, unsigned char Blinking);`**
  - Cette fonction initialise l'écran ainsi que les broches permettant de le contrôler. Les deux arguments permettent de choisir si le curseur doit être affiché (CursonOn) et, si oui, s'il doit clignoter (Blinking). Cette fonction doit être appelée avant toutes les autres. Un seul appel est nécessaire au début du programme.
- **`void LCD_Set_Cursor_Pos(unsigned char line, unsigned char col);`**
  - Déplace le curseur à une ligne et une colonne définies par les arguments
- **`void LCD_Write_Char(unsigned char data);`**
  - Ecrit un caractère à la place du curseur et déplace ledit curseur d'une position vers la droite. Notez qu'il n'y a pas de retour automatique à la ligne. Vous aurez besoin d'utiliser `LCD_Set_Cursor_Pos()` une fois la fin de ligne atteinte.
- **`void LCD_Write_HEX(unsigned char data);`**
  - Ecrit la valeur donnée sous forme hexadécimale (et affiche donc deux caractères au lieu d'un seul). Cette fonction utilise `LCD_Write_Char()`.
- **`void LCD_Write_String(char *str);`**
  - Affiche une chaine de caractères. Utilise `LCD_Write_Char()`. Ne gère toujours pas les fins de ligne et le retour à la ligne.
- **`void LCD_Clear();`**
  - Efface l'écran, et repositionne le curseur en haut à gauche (0, 0).
- **`LCD_HOME`**
  - Ceci n'est pas une fonction, mais une macro-commande. Elle repositionne le curseur en haut à gauche. Pour l'appeler, il suffit d'utiliser la ligne `LCD_HOME ;` (sans les parenthèses donc).

Afin d'être complet, il faut préciser que cette bibliothèque utilise le timer5 pour gérer les attentes de l'écran. Vous ne pouvez donc pas utiliser ce timer en parallèle.

a) Hello World (enfin!)

Ecrivez un programme qui dit bonjour au monde, juste pour vérifier que vous pouvez afficher du texte.

b) Affichage dynamique

Ecrivez un programme qui affiche un compteur qui s'incrémente chaque seconde. Pour éviter de passer trop de temps, nous nous contenterons d'un compteur 8 bits hexadécimal. Par contre, il faut l'afficher en bas à droite de l'écran.

Vous pouvez utiliser la fonction `DelayUs()` qui reçoit un entier *us* et qui occupe le processeur pendant environ *us* microsecondes (ce TP ne porte pas sur la maitrise du temps, inutile donc d'y consacrer trop de temps).

2) La liaison SPI

La liaison SPI est une liaison maître-esclave, série, synchrone, full-duplex.

- **Maître-esclave** : un composant (le maître) contrôle toutes les transmissions, les autres (les esclaves) ne peuvent que répondre aux requêtes du maître.
- **Série** : les bits de chaque valeur sont envoyés les uns après les autres sur le même fil.
- **Synchrone** : un signal d'horloge est transmis pour différencier les bits.
- **Full-duplex** : la communication se fait dans les deux sens en même temps (émission et réception de données) par opposition à half-duplex où la communication ne peut se faire que dans un sens à la fois.

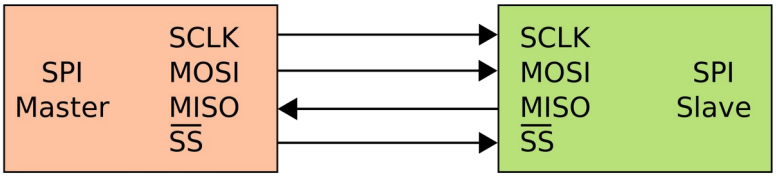

Figure 1: connection typique de deux composant en SPI (source Cburnet / Wikimedia)

Ce type de communication est prévu pour des liaisons locales (rarement plus de 10 à 15cm) à des vitesses de quelques dizaines de MHz. La liaison s'effectue sur quatre signaux : SCK, MOSI, MISO, SS (cf, fig. 1) :

- **SCK** : L'horloge de transfert. Cette horloge n'est pas périodique, elle n'est active que pendant les transferts.
- **MOSI** (Master Output, Slave Input) : les données circulant du maître vers tous les esclaves. Ce signal est également appelé SDI (au niveau des esclaves : Serial Data Input) ou SDO (au niveau du maître : Serial Data Output).
- **MISO** (Master Output, Slave Input) : la ligne de données relians la sortie de tous les esclaves vers l'entrée du maître. Cette organisation ne génère pas de court-circuit entre les sorties car chaque esclave est capable de gérer son impédance de sortie, et un seul esclave présente une faible impédence à un moment donné. Ce signal est également appelé SDO (au niveau des esclaves) ou SDI (au niveau du maître).

- $\overline{\mathbf{SS}}$ (Slave Select) : Ce signal est le seul signal qui n'est pas commun à l'ensemble des composants. Il existe un signal Slave Select par esclave. Le maître impose la valeur '1' sur tous les Slave Select, sauf sur celui qui le relie à l'esclave avec lequel il veut communiquer. Sur certains esclaves, ce signal peut également s'appeler $\overline{\text{EN}}$ (pour Enable)

Mise en œuvre de la liaison SPI :

Pour communiquer avec des composants sur une liaison SPI, il existe deux modules qui pilotent les broches et gèrent la communication pour que le processeur n'aie pas besoin de controler les signaux au coup d'horloge près. Il s'agit des modules SPI1 et SPI2. Le fonctionnement de ces modules est présenté rapidement au chapitre 18 (page 191) de la datasheet du microcontrôleur. Plus de détails sont disponibles dans la datasheet spécifique au module SPI sur le site des ressources de TP.

Nous allons communiquer avec une mémoire Flash qui est connectée sur les broches du module SPI1. Les registres concernés par le modules SPI1 sont SPI1CON, SPI1CON2, SPI1STAT, SPI1BUF et SPI1BRG. Le détail de ces registres est donné page 192, et chaque bit est expliqué dans les pages qui suivent. Quelques explication vous seront tout de même utiles :

- Le signal Slave Select sera géré indépendamment des transferts de données (comme une entrée/sortie normale).
- Le module gère le transfert complet. Pour envoyer une valeur, il suffit de l'écrire dans SPI1BUF. Pour consulter la dernière valeur reçue, il suffit de lire la valeur de SPI1BUF. Nous sommes en effet dans le cas où un registre présente un comportement différent en lecture et en écriture. A chaque fois qu'une nouvelle valeur est envoyée, une nouvelle valeur sera reçue. Cette valeur DOIT être lue avant le tranfert suivant pour éviter un blocage du module.
- Le registre SPI1BRG permet de définir la vitesse de l'horloge SCK en fonction de l'horloge des périphériques (120 MHz). La fréquence de l'horloge SCK est donnée par la formule (1)

$$F_{sck} = \frac{F_{Periph}}{2 \times (SPI1BRG + 1)} \qquad (1)$$

- SPI1STAT est un registre qui permet de récupérer différentes informations sur le fonctionnement du module. Seul le bit SPIBUSY nous sera utile.

En ce qui concerne les registres de configuration (SPI1CON et SPI1CON2), voici les quelques éléments qui vous permettront de le configurer :

- Pas de mode encadré (framed),
- L'horloge de base est l'horloge des périphériques (PBCLK)
- On n'utilise pas le mode buffer avancé. Bien que ce mode augmente les performances en débit sous certaines conditions, il complexifie la gestion des données.
- Les transferts de données se font sur 8 bits (imposé par la mémoire)
- Vue la valeur relativement faible de la fréquence de communication, le moment ou les bits sont lus en entrée a peu d'importance
- Les données en sortie doivent être écrites sur front descendant de l'horloge, la polarité de l'horloge (actif/inactif) n'a pas d'importance. Cela signifie que les bits CKE et CKP sont nécessairement de valeur inverse, mais qu'il n'y a pas d'importance sur lequel vaut '1'.
- Le microcontrôleur fonctionne en maître sur la communication (c'est lui qui initie les transferts).
- Nous utiliserons le module en scrutation.

Il ne reste plus qu'à configurer les entrées/sorties.

- Slave Select est connecté en F8 (bit 8 du port F)

- SCK sort sur la broche F6, cette connection est imposée par le routage du microcontrôleur.
- MOSI est en F2, plusieurs broches sont possible, c'est le concepteur de la carte qui a choisi cette connexion, il faut donc la configurer (cf p. 146)
- MISO est en F7, plusieurs broches sont possible, c'est le concepteur de la carte qui a choisi cette connexion, il faut donc la configurer (cf p. 142)

A ce stade, vous avez les ressources pour configurer le port de communication. Par contre, même si votre port fonctionne correctement, il reste compliqué de l'utiliser sans savoir comment fonctionne la mémoire Flash...

3) Fonctionnement de la mémoire Flash.

Au niveau physique, l'interface de la mémoire Flash correspond au standard SPI. Mais ce n'est pas suffisant pour définir l'usage du composant. An niveau logique, la communication de notre mémoire Flash est standardisée par le JEDEC (Joint Electron Device Engineering Council).

L'accès aux données se fait en envoyant des commandes, chacune ayant un format spécifique. Une remise à '1' du Slave Select termine une commande, alors qu'une mise à '0' précise que la prochaine valeur sur 8 bits sera un numéro de commande suivi de ses paramètres.
La mémoire est prévue pour fonctionner à la fréquence maximale de 50MHz.
Par exemple, la commande 0x9F demande à la mémoire d'envoyer ses identifiants. Une fois cette commande reçue, la mémoire envoie les données correspondant à ses propriété (0x9D pour identifier le fabricant, puis deux valeurs pour identifier le type de mémoire et sa capacité). Pour identifier la mémoire, la séquence est donc la suivante :

- Abaisser Slave Select
- envoyer 0x9F
- attendre que le transfert se termine
- envoyer une autre valeur (la valeur envoyée est sans importance, elle permet simplement d'activer l'horloge de transfert, et permet ainsi de recevoir la valeur que doit envoyer la mémoire)
- attendre que le transfert se termine
- lire dans le registre de réception (la valeur lue devrait être 0x9D)
- envoyer encore une autre valeur
- attendre que le transfert se termine
- lire dans le registre de réception (la valeur lue correspond au fabricant)
- envoyer une dernière valeur
- attendre que le transfert se termine
- lire dans le registre de réception (la valeur lue code la capacité mémoire)
- Relever Slave Select

Ecrire un programme qui affiche sur l'écran LCD les valeurs reçues par cette commande. Ce programme permet de valider que vous communiquez correctement avec la mémoire.

La datasheet de la mémoire Flash est également sur le site de ressources du TP. Sa lecture n'est cependant pas nécessaire, toutes les informations nécessaires sont fournies dans ce sujet.

Pour lire le contenu de la mémoire, il faut utiliser la commande 0x03 (read). Cette commande doit être suivie d'une adresse de départ de lecture sur 24 bits (3 octets, le poids fort en premier). Une fois les 4 octets de la commande envoyés, pour chaque octet envoyé, la mémoire retournera une donnée. La première donnée correspond à l'adresse envoyée (addr), la deuxième a la donnée

suivante (addr+1) et ainsi de suite. Il est ansi possible de lire tout le contenu de la mémoire en une seule commande. Pour terminer la commande de lecture, il suffit de remonter le Slave Select.

Un message de 32 caractères est stocké à partir de l'adresse 0x000000 sur la mémoire de chaque carte. Affichez ce message sur l'écran (16 caractères par ligne).

4) Utilisation avancée (facultatif)

Pour simplifier l'usage des modules SPI, on ne parle pas de fonctions d'envoi ou de réception, mais de fonction d'échange. En effet, il faut toujours envoyer quelque chose pour recevoir des données. Pour simplifier votre code, vous pouvez donc écrire une telle fonction.

Les modules SPI du pic32mx370 contiennent une mémoire intégrée capable de mémoriser une file d'attente de 16 octets dans chaque sens (envoi et réception). Modifiez votre configuration pour utiliser ces buffers (*Enhanced buffer* dans la datasheet). L'usage de ces buffers n'augmente pas la vitesse de transfert, par contre, il minimise le temps d'attente entre deux envois successifs et participe donc à l'augmentation des performances globales du système.