

Polytech Clermont – Ferrand
Département Génie Electrique 5A

Structure d'un programme de transfert USB <-> SPI

Note d'application

Table des matières



Introduction.....	3
1. Fonctionnement du bus SPI.....	4
a. Topologie du bus SPI	
b. Méthode de transmission entre un maître et un esclave	
c. Programmation sur microcontrôleur	
2. Fonctionnement du bus USB.....	6
a. Protocole de réception de donnée	
b. Programmation sur microcontrôleur	
Conclusion.....	8

Introduction

Dans le cadre des projets de fin d'études de génie électrique à Polytech Clermont-Ferrand, le département Génie Civil et le laboratoire LIRMM (Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier) nous a donné la possibilité de travailler sur une problématique concernant la réception des informations sur la déformation de certains matériaux (pour le département Génie Civil) et des pattes du robot hexapode RHEX (développé par LIRMM) . Cette note d'application abordera seulement la structure de transmission de cette déformation en utilisant les protocoles de communication USB et SPI.

La demande du projet s'est donc présentée pour deux clients qui avaient certains points communs dans leurs cahiers de charges : Récupérer les données de déformation à partir d'un convertisseur analogique qui lui convertit les informations des capteurs fixés d'une part sur les matériaux à étudier comme l'acier, d'autre part sur les pattes du Robot RHEX. Ces informations seront accessibles sur un serveur, à partir d'un ordinateur on pourra donc observer l'évolution des déformations suite aux efforts appliqués sur les points d'application.

Nous allons donc aborder la méthode de transfert, le programme en lui-même qui va permettre ce transfert de données.

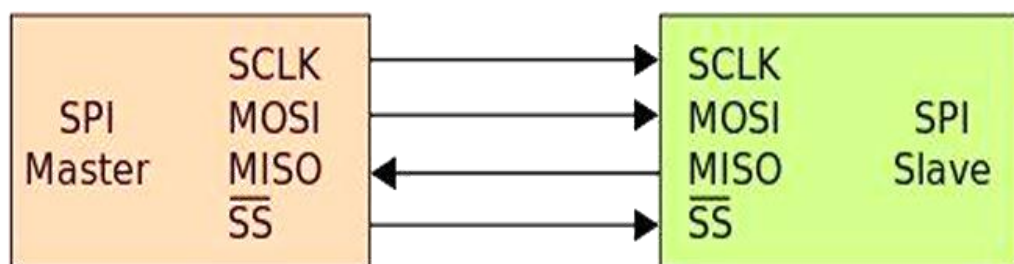
I. Fonctionnement de la communication SPI

a. Topologie du SPI :

Une liaison SPI signifie Serial Peripheral Interface. C'est un bus de données série synchrone, qui opère en mode Full-duplex. Les circuits communiquent selon un schéma maître-esclaves, où le maître s'occupe totalement de la communication. Plusieurs esclaves peuvent coexister sur un même bus, dans ce cas, la sélection du destinataire se fait par une ligne dédiée entre le maître et l'esclave appelée « chip select ».

b. Méthode de transmission entre un maître et un esclave :

Il existe deux modes de fonctionnement SPI, mais pour notre cas nous allons nous intéresser au mode Maître (PIC18F4550) – Esclave (Convertisseur analogique numérique). Voir figure ci-dessous.



En mode Master, le PIC contrôle l'horloge (la broche SCLK).

- Une écriture du PIC (maître) provoque 8 impulsions sur SCLK et l'envoi de la donnée sur MOSI.

- Une lecture de l'ADC n'est terminée que quand les 8 impulsions sur SCLK sont terminées.
- Le PIC envoie une commande via la broche MOSI et il reçoit la réponse de l'ADC via la broche MISO. C'est donc un jeu de lecture-écriture qui va nous permettre de récupérer les informations de conversions de déformation.

c. Programmation sur microcontrôleur :

Pour le programme de communication SPI entre le convertisseur et le microcontrôleur, il faut d'abord effectuer **l'initialisation** des registres du microcontrôleur PIC18F4550 et ceux de l'ADC, ensuite une partie **d'acquisition** de données converties. Cette programmation est faite sur le logiciel fourni par Microchip : MPLAB.



Pour charger ce programme sur notre carte de traitement, nous avons utilisé le Bootloader : Pickit3.



➤ **Initialisation :**

Registres du PIC18F4550 :

Les registres ADCON1, SSPCON1, SSPSTAT ont été mis à jour suivant la documentation technique pour établir la communication SPI. De plus, nous avons fixé les ports d'entrée et de sortie sur le microcontrôleur avec les registres TRISA, TRISB et TRISC.

```
ADCON1 |= 0xF ;           //Activate digital pin
PORTAbits.RA5 = 0;
```

```

TRISC = 0x00;          // Set All on PORTC as Output
TRISA = 0x00;          // Set all on PORTA as Output
TRISB = 0x00;          // Set all on PORTB as Output

TRISAbits.TRISA5 = 0;  // RA5/SS - Output (Chip Select)
TRISCbits.TRISC7= 0;   // RC7/SDO - Output (Serial Data Out)
TRISBbits.TRISB0= 1;   // RB0/SDI - Input (Serial Data In)
TRISBbits.TRISB1= 0;   // RB1/SCK - Output (Clock)
TRISDbits.TRISD1 = 0;

SSPSTATbits.CKE = 0;    // Set SMP=0 and CKE=1. Notes: The lower 6 bit is read
                        only
SSPSTATbits.SMP = 0;
SSPCON1bits.CKP = 1 ;
SSPCON1bits.SSPEN = 1;  // enable the SPI communication
SSPCON1bits.SSPM=0x02;

```

Registres de l'ADC ad7176 :

Registre de communication : COMMS. Il est en écriture et doit être initialisé à chaque communication.

Le bit WEN barre doit être mis à l'état bas pour commencer toute forme de communication. Le bit R/W (0 : en mode écriture, 1 : en mode lecture). La séquence RA détermine quel registre sur lequel on communique avec.

Status register : registre en mode lecture, il permet de connaître l'état du convertisseur.

RDY : bit qui va déterminer s'il y'a une nouvelle donnée disponible et s'il est en attente d'une donnée.

CHANNEL : permet de lire sur quelle chaîne de conversion on est exactement.

ADC MODE REGISTER (ADC MODE) Registre qui contrôle le mode dans lequel on travaille. Il est en mode écriture et lecture. "Pour notre convertisseur on est en mode Single Conversion Mode. Le delay est à 0. Le bit SING_CYC doit est mis à 1 parce qu'on utilise qu'une seule chaîne de conversion.

Serial Interface Register : Il permet de configurer les différentes options d'interface.

GPIO Configuration : Ce registre contrôle l'usage général broches E / S de l'ADC.

➤ Acquisition :

On va lire sur un registre du microcontrôleur appelé SSPBUF qui lui stocke l'information du registre Data Register de l'ADC.

Data Register : Le registre de donnée contient le résultat de la conversion de l'ADC. 24 bits de données. La lecture du registre de donnée apporte le bit RDY, et la broche à l'état haut si elle était à l'état bas. L'ADC ne pourra pas lire une autre donnée si le registre de donnée est en pleine lecture.

II. Fonctionnement du bus USB :

a. Protocole de réception de données :

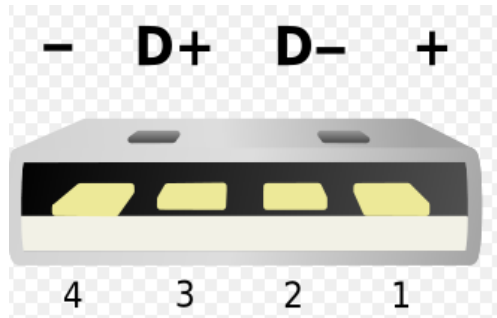
Lorsque les données sont récupérées il faut les envoyer au serveur et pour cela nous avons choisi d'utiliser un mini PC appelé Raspberry Pi qui va contrôler le PIC en tant que maître de ce dernier. Il va lui envoyer des demandes de récupération des données converties.

Pour communiquer entre la Raspberry Pi et la carte microcontrôleur qui aura reçu les données converties, nous avons choisi d'utiliser le protocole USB (Universal Serial Bus). Microchip fournit déjà un code de base permettant d'utiliser le protocole USB sur certain microcontrôleur et on a pu trouver celui de notre microcontrôleur : PIC18F4550.



Raspberry.

Ce protocole permet de faire communiquer deux composants à travers 4 fils : VCC, VDD, D+ et D-. Les données transitent en passant par les broches D+ et D-.



b. Programmation du microcontrôleur :

Pour le transfert des données, il y'a une autre partie du programme qui va donc récupérer la valeur contenue dans le registre SSPBUF. Cette partie du programme est fournie par Microchip. On a juste à connecter notre Raspberry à la carte en USB et signifier l'adresse du port de la Raspberry à notre routine de traitement.

Cette partie du programme est appelée **traitement**.

Conclusion :

Suite à ce travail, je peux dire que j'ai mieux compris le rôle de certains composants électroniques. Aussi le choix de tel ou tel type de communication pour le transfert de données est conforme à un cahier de charge donné. De plus la lecture de documentation technique pour comprendre comment fonctionnent un composant électronique.