**Notes d’application**

**Partie commande**

**Etudiant :DALIL ESSAKALI Youssef**

**1-Module SPI**

Les modules SPI des dsPIC leur permettent de communiquer avec des périphériques en utilisant l’interface SPI.

**Présentation**

Les microcontrôleurs les mieux fournis en périphériques de la famille dsPIC30F disposent de deux modules SPI distincts. Dans la suite de cette page, x représente toujours le numéro du module concerné (1 ou 2).

Chaque module SPI permet de disposer d’une interface SPI complète.

Dans le guide d’utilisation de la famille dsPIC30F, le chapitre concernant les modules SPI s’étend des pages 525 à 547. Consultez-le pour disposer de toutes les informations disponibles.

**Registres et broches**

Le module SPI utilise trois registres : SPIxBUF, SPIxCON et SPIxSTAT.

Le registre SPIxBUF (SPI x BUFfer) est le registre de donnée du module SPI. En fait, deux registres se cachent derrière ce nom : SPIxTXB et SPIxRXB (SPI x Transmit / Receive Buffer), qui sont les registres d’émission et de réception. Quand on lit SPIxBUF, c’est la valeur de SPIxRXB qu’on obtient. Quand on écrit dans SPIxBUF, c’est le registre SPIxTXB qu’on modifie. On utilise uniquement SPIxBUF, qui est en quelque sorte un « raccourci ».

Le registre SPIxCON (SPI x CONfig) permet la configuration du module, et SPIxSTAT (SPI x STATus) permet aussi de connaitre l’état du module.

Le module peut être utilisé en mode 3 ou 4 broches. En mode 3 broches, seule les broches SDIx, SDOx et SCKx (correspondant aux broches DIN, DOUT et SCKL de l’interface SPI) sont utilisée. En mode 4 broches, la broche SSx (CS de l’interface) est utilisée en plus des trois autres.

**Activation du module**

Pour valider le fonctionnement du module SPI, il faut mettre à 1 le bit SPIEN (SPI ENable) du registre SPIxSTAT. Quand le module SPI est activé, les quatre broches SDIx, SDOx, SCKx et SSx sont pilotées par le module.

**Mise en veille**

Si le bit SPISIDL (SPI Stop IDLe) du registre SPIxSTAT est mis à 1, le module SPI stoppera ses activités lorsque le dsPIC entre en mode veille. Sinon, le module continuera de fonctionner même en mode veille.

**Désactivation de SDOx**

La broche SDOx peut être désactivée. Seul le récepteur du module SPI fonctionnera. La broche SDO pourra être utilisée par un autre périphérique ou comme port d’E/S. Pour désactiver cette broche, il faut mettre à 1 le bit DISSDO du registre SPIxCON (DISable SDO). Si ce bit est laissé à 0, la broche SDO sera utilisée par le module.

**8 / 16 bits**

Les modules SPI des dsPIC peuvent être paramétrés pour communiquer en mode 8 ou 16 bits. Pour sélectionner le mode 16 bits, mettez à 1 le bit MODE16 du registre SPIxCON. Pour transférer des octets, laissez-le à 0.

**Lecture des bits**

La lecture d’un bit peut s’effectuer au milieu ou à la fin du bit. Pour lire les données au milieu des bits, il faut mettre le bit SMP (SaMple Phase) du registre SPIxCON à 0. Pour les lire en fin de bits, mettez-le à 1.

Si le module est paramétré en esclave (voir plus bas), ce bit doit être laissé à 0.

**Polarité de l’horloge**

La polarité de l’horloge peut être sélectionnée au moyen du bit CKP (ClocK Polarity) du registre SPIxCON. Ce bit indique l’état de l’horloge au repos. L'état de repos correspond à l'état de l'horloge pendant les periodes de non-transmission. L'autre état est appelé état actif. Par exemple, si CKP est à 1, l’horloge sera au niveau logique 1 hors des périodes de transmission.

**Lecture des bits**

Les bits peuvent être lus sur SDIx quand l’horloge passe de l’état de repos à l’état actif, ou quand elle passe de l’état actif à l’état de repos. Dans le premier cas, le bit SMP (SaMple Phase) du registre SPIxCON doit être mis à 1. Dans le second cas, ce bit doit être mis à 0.

**Ecriture des bits**

Les bits peuvent être écrits sur SDOxquand l’horloge passe de l’état de repos à l’état actif, ou quand elle passe de l’état actif à l’état de repos. Dans le premier cas, le bit CKE (ClocK Edge) du registre SPIxCON doit être mis à 0. Dans le second cas, ce bit doit être mis à 1.

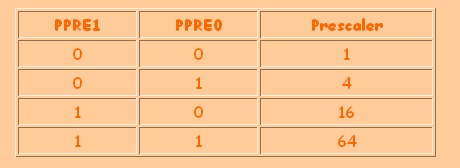
**Maitre / Esclave**

Pour paramétrer le module SPI en maitre, il faut mettre le bit MSTEN (MaSTer ENable) du registre SPIxCON à 1. S’il est mis à 0, le module sera paramétré en esclave.

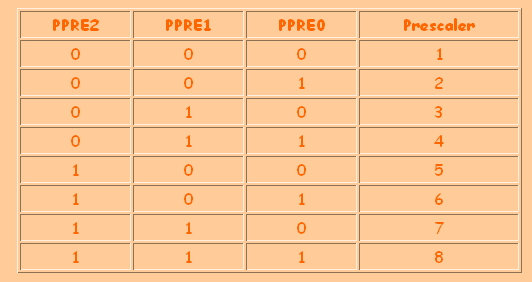
**Vitesse de transmission**

En mode maitre, on peut définir la vitesse de transmission du module SPI au moyen de deux prescaler.

Le prescaler 1 est paramétré par les bits PPRE1 et PPRE0 (Primary PREscaler) du registre SPIxCON :



Le prescaler 1 est paramétré par les bits SPRE2, SPRE1 et SPRE0 (Secondary PREscaler) du registre SPIxCON :



La fréquence de transmission s’obtient en divisant la fréquence d’horloge système par le produit des deux prescalers.

**Etat du module**

Le bit SPIROV (SPI Receive OVerflow) du registre SPIxSTAT est mis à un quand un caractère est reçu alors que le buffer de réception est plein. Le buffer de réception à une profondeur de 1 caractère.

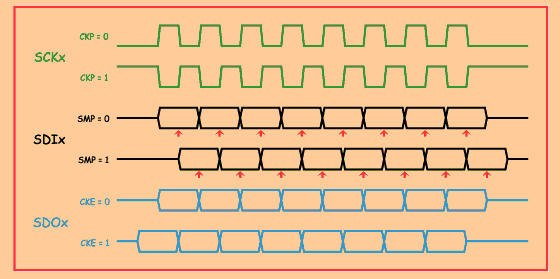
Le bit SPITBF (SPI Transmit Buffer Full) du registre SPIxSTAT est mis à un quand le buffer d’émission (1 caractère) est plein. Avant d’envoyer un caractère, il faut donc s’assurer que ce bit est à 0.

Le bit SPIRBF (SPI Receive Buffer Full) du registre SPIxSTAT est mis à un quand le buffer de réception est plein. Il est donc inutile de lire le buffer de réception si ce bit est à 0.

**NB :** Pour le dspic30f4011, les broches SPI coincident avec les broches de l’ICD2, ce qui rend l’émission et la réception des données en mode débugger impossible. On est donc obligé de passer en mode programmer pour vérifier le bon fonctionnement de l’émission et de la réception du module SPI.

**Mode de transmission**

Le paramétrage des bits CKP, CKE et SMP autorise 8 modes de fonctionnement (module en maitre). Le chronogramme ci-dessous montre le comportement des trois lignes SDIx, SDOx et SCKx en fonction du paramétrage de ces trois bits :



Les flèches rouges représentent le moment ou le dsPIC effectue ses samples, c'est-à-dire le moment où il lit les données sur SDIx.

**Interruption**

L’émission et la réception s’effectuant en même temps (voir chapitre sur les interfaces), le module SPI ne génère qu’une interruption, en fin de transmission. Cette interruption s’appelle SPIxInterrupt.

**NB**: L’horloge n’est générée que lorsqu’il y a une transmission de données. Or, dans notre cas, le but est de recevoir la donnée de l’ad2s90. Pour cela, on émet des données fictives pour générer l’horloge qui cadencera la réception des données. Ceci est possible puisque l’émission et la réception se font en même temps.

**Exemple d’une fonction de Lecture SPI :**

unsigned char SPIRead(void)

{

unsigned char data;

int tempo=0;

PORTBbits.RB2=0; // Activation de notre chip AD2S90

tempo=1; // Attente que le chip soit bien actif

SPI1BUF = 0xFFFF; // Ecriture d'une donnée fictive pour générer l'horloge qui

//cadencera la réception de l'angle o.

while(SPI1STATbits.SPITBF); // Fin de la transmission = Fin de la réception

data = SPI1BUF; // la donnée reçue est stockée

SPI1STATbits.SPIROV = 0;

return data;

}

**2-Module PWM:**

C’est un module qui est trés utilisé dans les applications de commande de moteur.

Dépendant du dspic30f choisi, on peut avoir jusqu’à 8 sorties PWM. Dans notre cas (dspic30f4011), on n’a que 6 sorties PWM dont 3 seulement seront utilisées, et les autres seront considérées comme étant des simples entrées/sorties du dspic30f.

**Registre OVDCON :**

Le registre permettant de configurer les sorties PWM est le registre OVDCON. C’est donc le registre sur lequel il faudra agir pour controler les commutations entre les phases du moteur, il doit étre initialisé à 0 (Moteur à l’arrét).

**Registres PDCx :**

Le rapport cyclique des signaux PWM dépend essentiellement de la consigne (pédale), en fonction de laquelle on met à jour les nouveaux rapports cycliques des signaux PWM..

Les registres responsables sur les rapports cycliques sont les registres suivants : PDC1, PDC2, PDC3.

PDCx agit sur le rapport cyclique des deux signaux PWMxH et PWMxL (désactivés).

Les PDCx seront mis à 0 lors de l’initialisation, puisque le moteur sera encore à l’arrét.

**Registre PTPER :**

Ce registre permettant de contrôler la période des signaux PWM.

Pour avoir une période FPWM, avec fcy est la fréquence de fonctionnement de notre microcontrôleur, le constructeur impose d’utiliser la relation suivante:

**PTPER=Fcy/FPWM-1**

**Registre PWMCON1 :**

Ce registre permet de configurer les sorties du module PWM en sorties PWMs ou entrées/sorties du dspic30f. Il permet également de choisir la complémentarité ou pas des sorties PWMxH et PWMxL.

Dans notre cas, ce registre prendra la valeur 0x0770, c’est une valeur qui fait des sorties PWMxL des entrées/sorties normales, alors que les PWMxH sont des sorties PWM.

On n’a de complémentarité entre les sorties PWM, ce qui fait qu’on n’a pas besoin dans notre cas de gérer les temps morts à l’aides des registres **DTCON1** et **DTCON2.**

**Registre PTCON**

Ce registre nous permet d’activer le timer associé au module PWM, il nous permet également de choisir le mode de fonctionnement PWM. Le mode choisi dans notre cas est le free running mode.

**Ci-dessous un exemple de fonction d’initialisation du module PWM :**

void initPWM (void)

{

PTPER=Fcy/FPWM -1 ; // Période des PWM 16khz

OVDCON=0x0000; // C'est le registre qui nous permet de controler les sorties PWM

//(La commutation entre phases de semi-conducteurs du haut est

//faite en changeant la valeur de OVDCON)

PWMCON1=0x0770; // Registre d'initialisation du module PWM

PWMCON2=0x0100; // Registre d'initialisation du module PWM

PDC1=0; // Initialisation des rapports cycliques PWM à 0

PDC2=0;

PDC3=0;

//IFS2bits.PWMIF = 0;

//IFS2bits.FLTAIF =0;

PTCON=0x8000;

}

**Ci-dessous deux chronogrammes avec des rapports cycliques maximum et minimum :**

