EXERCICES SPI

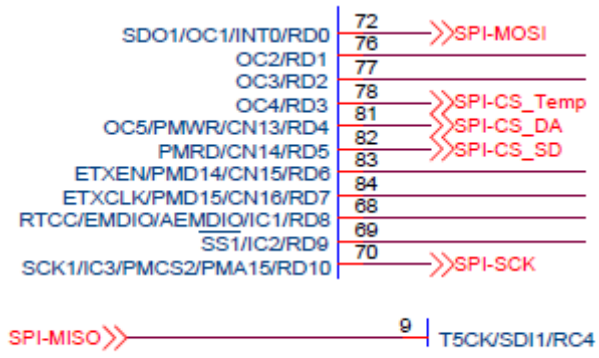
OBJECTIFS

Ces exercices pratiques ont pour buts de réaliser une communication par le bus SPI entre le PIC32 et un périphérique, puis entre 2 kits PIC32MX, un étant master et l'autre slave. L'objectif est de découvrir le fonctionnement en modes master et slave ainsi que la gestion de l'interruption du SPI.

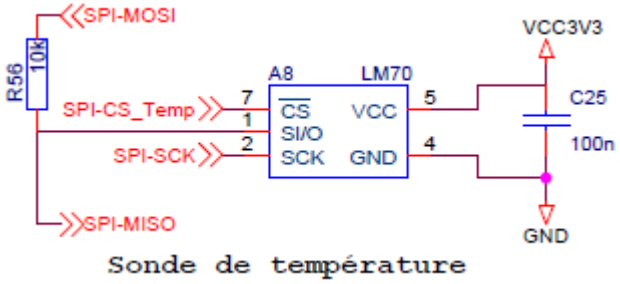
EXERCICE 8-1 : SPI SIMPLE

Il s'agit de dialoguer avec le capteur de température LM70 présent sur votre kit.

PIC32 :

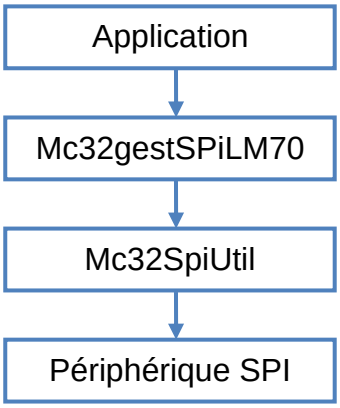


LM70 :



Librairie à disposition

Utilisez la librairie Mc32gestSPiLM70 à disposition dans :
...\Maitres-Eleves\SLO\Modules\SL229_MINF\PIC32MX_Utilitaires(PlibHarmony)\SPI
Cette librairie utilise elle-même les fonctions disponibles dans Mc32SpiUtil.



Réalisation et cahier des charges

- A l'aide de la librairie mise à disposition, lisez la température du LM70 à intervalle régulier (20 ms).
- Affichez-la sur le LCD.

Indications

- Utilisation d'un timer avec une interruption de période de 20 ms.
- Implémentation de la classique fonction APP_UpdateState().
- Lecture et affichage dans APP_STATE_SERVICE_TASKS.

Durée approximative

½ p.

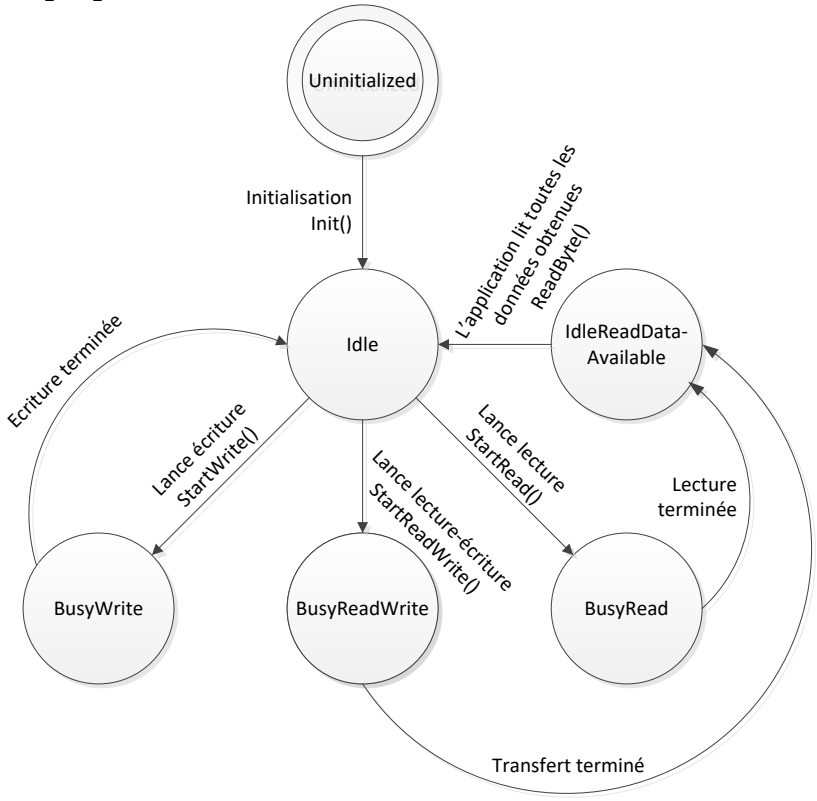
Question subsidiaire

Procédez à l'affichage de la température au ¼ de degré sans recours au type float.

EXERCICE 8-2 : SPI PAR MACHINE D'ETAT

Les fonctions de la librairie Mc32gestSPiLM70 sont bloquantes tant que la communication SPI n'est pas terminée.
Il s'agit de remplacer cette librairie par une version plus élaborée fonctionnant sur le principe d'une machine d'état qui autorisera l'utilisation de fonctions non bloquantes.

Machine d'état proposée



Fonctions proposées

- `void SPI_Init(void);` — Fonction à appeler 1x au démarrage pour init.
- `void SPI_StartWrite(uint32_t nBytes, uint8_t* pBytesToWrite);` — Ecriture. Comme le SPI est obligatoirement full-duplex, les données reçues ne seront pas traitées
- `void SPI_StartReadWrite(uint32_t nBytes, uint8_t* pBytesToWrite);` — Lecture/écriture. Comme le SPI est obligatoirement full-duplex, des données sont reçues simultanément à l'envoi
- `void SPI_StartRead(uint32_t nBytes);` — Lecture. Comme le SPI est obligatoirement full-duplex, il faut envoyer des données "bidons" pour faire une lecture
- `SPI_STATES SPI_GetState (void);` — Pour obtenir l'état interne de la SM SPI.

- `uint8_t SPI_ReadByte(void);` Lecture d'un byte dans buffer réception.
- `void SPI_DoTasks(void);` Fonction à appeler périodiquement pour gestion SPI.

Réalisation et cahier des charges

- Réalisez le dialogue SPI sous forme de machine d'état selon l'architecture proposée ci-dessus. Aucun appel ne doit être bloquant.
- Lisez la température du LM70 à intervalle régulier (20 ms).
- Lorsque la transaction SPI de lecture est terminée, affichez la température sur le LCD.

Indications

- Un canevas de librairie vous est fourni dans les fichiers spi_sm.c et .h
- La fonction SPI_DoTasks() doit être appelée en tâche de fond. Cela peut être fait par polling dans la fonction APP_Tasks() et pas nécessairement par interruption.
- Le traitement dans APP_STATE_SERVICE_TASKS consistera à :
  - Lancer une lecture.
  - Contrôler si une valeur est disponible et l'afficher le cas échéant.
- Vous devrez gérer le signal "slave select" manuellement via software.

Observation du fonctionnement

Il est demandé d'observer une transaction SPI complète (de l'activation de /SS jusqu'à sa désactivation) :

canal 1 : /SS canal 2 : SCK canal 3 : SDO canal 4: SDI	

Durée approximative

2 p.

EXERCICE 8-3 : SPI MASTER & SLAVE

Il s'agit de faire communiquer 2 kits entre eux, l'un étant le master SPI et l'autre le slave.

Connexions

Utilisation du SPI 4 pour le master et le slave (signaux câblés sur le XBee). Le master génère /SS4 et le clock. Croisement des lignes SDO et SDI.

Master			Slave	
signal	pin PIC32		pin PIC32	signal
/SS4	40	→	40	/SS4
SCK4	39	→	39	SCK4
SDO4	50	→	49	SDI4
SDI4	49	←	50	SDO4
GND		—		GND

Trame Master → Slave

La trame émise par le maître correspond à :

- Envoi d'une valeur 16 bits pour le DAC slave.
- Suivi de la zone de réponse pour les 2 valeurs des 2 canaux du convertisseur AD du slave. Les valeurs des 4 octets servent à la réponse du slave et permettent également l'observation.

		Lsb	Msb	Lsb	Msb
Lsb DAC	Msb DAC	0x01	0x11	0x02	0x22

Trame Slave → Master

Le SPI étant full-duplex de par nature et fonctionnant sur le principe d'un décalage de bits, la trame de réponse du slave aura nécessairement la même taille que la trame du master.

La trame de réponse du slave correspond à :

- Début par 2 valeurs fixes.
- Suivi des 4 octets des valeurs brutes des 2 canaux de l'AD.

Lsb	Msb				
0x81	0xC3	Lsb AD0	Msb AD0	Lsb AD1	Msb AD1

Réalisation et cahier des charges master

- Reprise des fonctions master de l'exercice 8-2.
- Transposition du périphérique SPI1 au SPI4.
- Toutes les 20 ms, le master :
  - Initie une transaction SPI selon le format ci-dessus. La valeur à envoyer au DAC slave peut être issue d'un potentiomètre par exemple. La valeur envoyée est affichée.
  - Contrôler si une trame a été reçue. Les valeurs reçues sont affichées le cas échéant.
- Affichage :

```
Master SPI / <nom>
TX : <valeur>
RX : <val.1> / <val.2>
<ligne de statut>
```

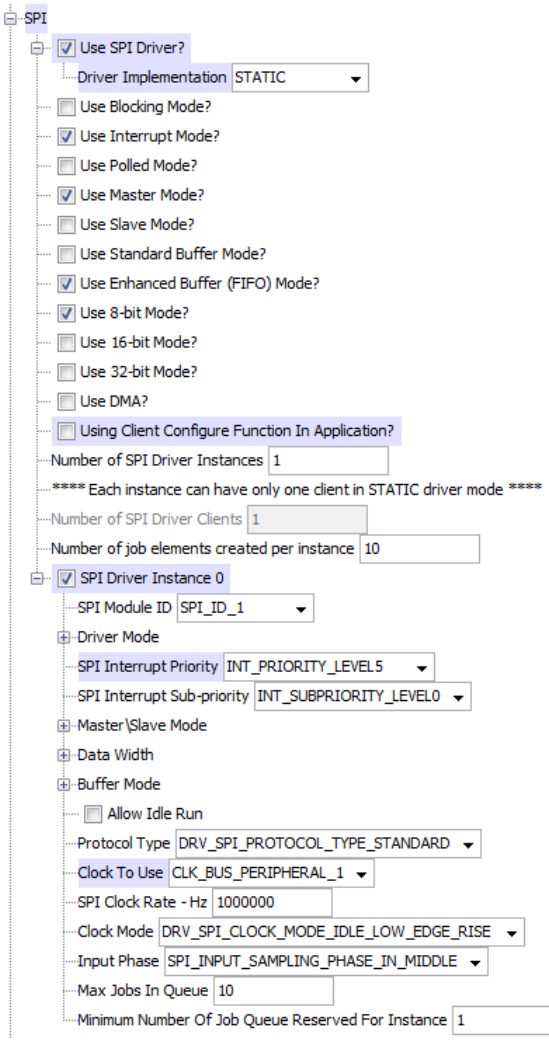
Réalisation et cahier des charges slave

- Toutes les 20 ms, le slave :
  - Lit ses entrées AD et affiche les valeurs.
  - Contrôle si une nouvelle valeur DAC a été reçue, l'applique au DAC et l'affiche le cas échéant.
- Via interruption SPI RX :
  - Dès réception du premier octet d'une trame, chargement des valeurs AD à transmettre dans le buffer SPI (4 derniers octets de la trame).
  - Dès réception d'une trame complète :
    - Décodage et transmission de la nouvelle valeur DAC à l'application.
    - Pré-chargement du buffer SPI avec les 2 premiers octets du prochain transfert (0x81 et 0xC3).
- Affichage :

```
Slave SPI / <nom>
TX : <val.1> / <val.2>
RX : <valeur>
<ligne de statut>
```

Indications

- Pour la configuration du SPI slave, il est possible de s'inspirer de cette configuration master (il faut cocher "slave mode" à la place de "master mode").



- La gestion de l'entrée /SS est automatique par le périphérique SPI.
- Remplacer la gestion des interruptions SPI générée par le code simplifié ci-dessous :

```
//interrupt SPI
void __ISR(_SPI_4_VECTOR, ipl5AUTO) _IntHandlerSPIInstance0(void)
{
    //interrupt SPI TX ?
    if (SYS_INT_SourceIsEnabled(INT_SOURCE_SPI_4_TRANSMIT) &&
SYS_INT_SourceStatusGet(INT_SOURCE_SPI_4_TRANSMIT))
    {
        SYS_INT_SourceStatusClear(INT_SOURCE_SPI_4_TRANSMIT);
        //ne devrait pas arriver ! (interrupt tx désactivée)
    }

    //interrupt SPI RX ?
    if (SYS_INT_SourceIsEnabled(INT_SOURCE_SPI_4_RECEIVE) &&
SYS_INT_SourceStatusGet(INT_SOURCE_SPI_4_RECEIVE))
    {

        //vider d'abord buffer rx, puis quittancer interrupt
        SYS_INT_SourceStatusClear(INT_SOURCE_SPI_4_RECEIVE);

        //suite traitement...

    }

    //interrupt SPI Error ?
    if (SYS_INT_SourceIsEnabled(INT_SOURCE_SPI_4_ERROR) &&
SYS_INT_SourceStatusGet(INT_SOURCE_SPI_4_ERROR))
    {
        SYS_INT_SourceStatusClear(INT_SOURCE_SPI_4_ERROR);

        //traitement erreur...
    }
}
```

- Mise en route du slave et du master : Pour assurer un bon démarrage, il faut maintenir en reset le master, effectuer un reset du slave, puis relâcher le reset du master.
- Détails de la réalisation : Réalisation de l'exercice par groupe de deux. Un des étudiants réalise le maître et l'autre l'esclave.

Observation du fonctionnement
Il est demandé d'observer une transaction SPI complète (de l'activation de /SS jusqu'à sa désactivation) :

canal 1 : /SS canal 2 : SCK canal 3 : SDO (master) canal 4: SDI (master)	

Durée approximative
3 p.

Question subsidiaire
Compte tenu du temps de réaction du slave (à la réception du premier octet, le slave doit mettre immédiatement dans le buffer d'émission les 4 octets des valeurs des entrées AD), quelle est la fréquence maximale d'horloge du SPI ?
Il est demandé d'observer le temps de réaction entre l'arrivée des octets et leur traitement dans l'interruption SPI (par exemple avec un 1 marqueur pour la durée de l'interruption SPI).

canal 1 : /SS canal 2 : SCK canal 3 : SDI (data slave → master) canal 4: LED1 (marqueur interruption SPI)	