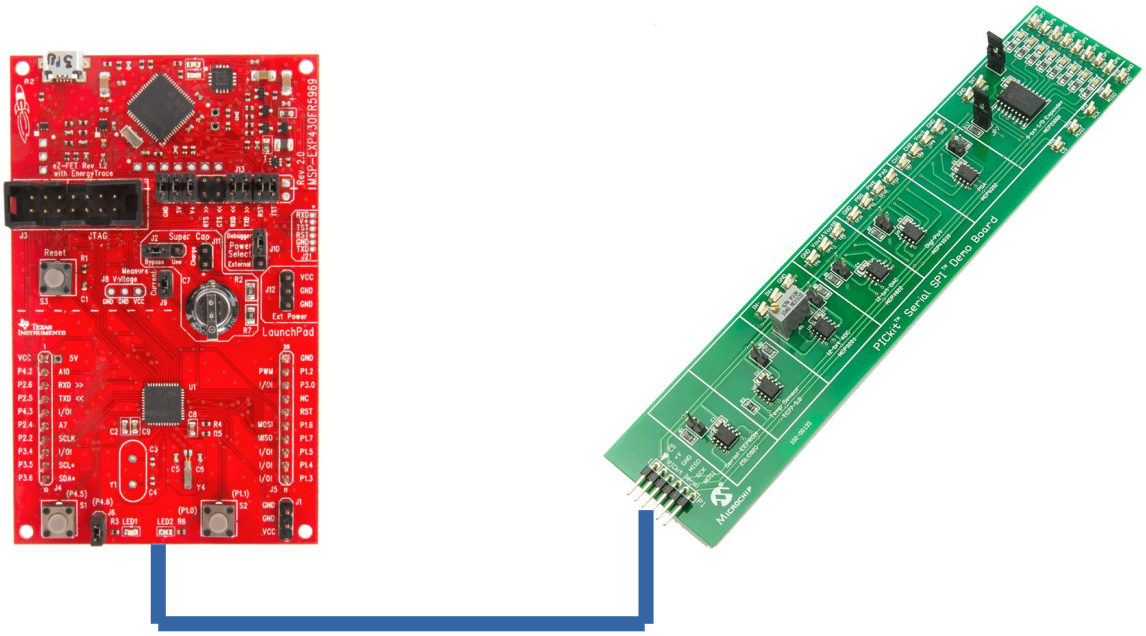
Microprocesseurs

Programmation en C pour MSP430FR5969



TP6: Interface SPI

Professeurs: D. Bullot / P. Buccella

Nom des étudiants:

Date du laboratoire:

Table des matières

1 Objectifs

Les objectifs de ce travail pratique sont:

- réaliser avec le Launchpad MSP430FR5969 une mesure de température avec d'un capteur de température sur le PICKit qui possède un interface sériel SPI. Le capteur est en mode Slave et le microcontrôleur en mode Master.
- Utilisations des périphériques **USCI** et GPIO du MSP430.
- Utilisation de quelques éléments de base du langage C tel que tableau, procédures et fonctions, passage de paramètres par valeur et référence, pointeurs.
- utilisation de ***printf*** pour visualiser des résultats sur la console CCS.
- Utilisation de l'outil de développement Code Composer Studio.

2 Résultat demandé

a) Rendu à la fin des deux premières périodes de la séance de TP de/des logigramme/s qui représentent le code à élaborer ;

b) Remise à la fin du TP du code principal implémenté de manière à respecter au mieux les spécifications et incluant tous les commentaires utiles.

Pour exporter le projet, voir :
https://software-dl.ti.com/ccs/esd/documents/ccs_sharing-projects.html

Notes :1

- 1 fichier zip à charger sur Teams avec le projet CCS du TP.
- 1 fichier en pdf avec les logigrammes.

Les fichiers sont à nommer comme suit : TP6_Prénom1_Nom1_Prénom2_Nom2.zip et TP6_Prénom1_Nom1_Prénom2_Nom2.pdf

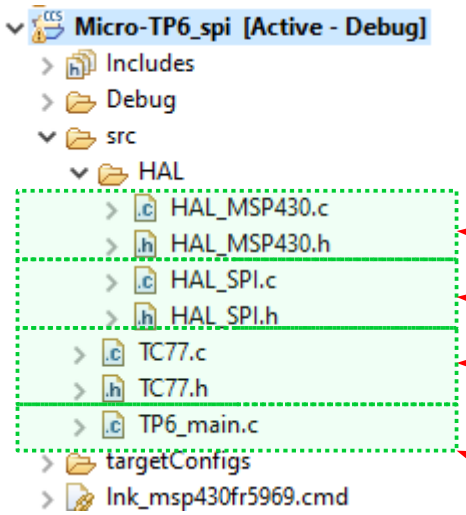
3 Informations complémentaires

3.1 Documents de référence

Les documents utiles pour ce TP sont:

- Document «C-Programming_Style_and_coding conventions.pdf»
- Polycopié/exercices/notes de cours, labos précédents, spécialement
  - 04_Langage-C_Concepts
  - 04_Langage-C_Functions
  - 05_Périphériques-MSP430-GPIO
  - 05_Périphériques-MSP430-USCI-SPI
- les fichiers headers CCS avec les informations d'adresses et bits qui se trouvent dans le répertoire de l'installation de CCS
  - msp430fr5969.h
- Le fichier du projet Micro-TP6.zip sur Moodle
- Data sheets:
  - Schéma PICKit: *51658a-1 - PICkit Serial SPI Demo Board User's Guide.pdf*
  - *Data sheet TC 77: 20092a – TC77.pdf*
  - *Data sheet MCP23S08: 21919e - Microchip MCP23S08.pdf*

3.2 Layout du projet CCS Micro-TP6



HAL – Hardware Abstraction Layer pour MSP430 – **partie 2)** GPIO à initialiser

HAL – Hardware Abstraction Layer pour SPI – **partie 1)** SPI à initialiser

Implémentation interface TC77
partie 3) – lecture du TC77 par SPI

Parties 4-5) conversion de la valeur de température lue du TC77 et filtrage de ceci

3.3 Capteur de température TC77

Le capteur de température peut être décrit à l'aide de son schéma bloc. Il s'agit d'une mesure de température par l'intermédiaire de la variation des caractéristiques d'une diode.

Un convertisseur Sigma-Delta dont la fréquence d'échantillonnage est de 30kHz permet d'obtenir une résolution de 13 bits. Un interface MICROWIRE compatible SPI permet d'atteindre deux registres. L'un pour la configuration du système (mode de travail, mode basse consommation) ainsi que le contrôle du circuit en phase de test et de calibration. Le second registre contient la valeur de la mesure.

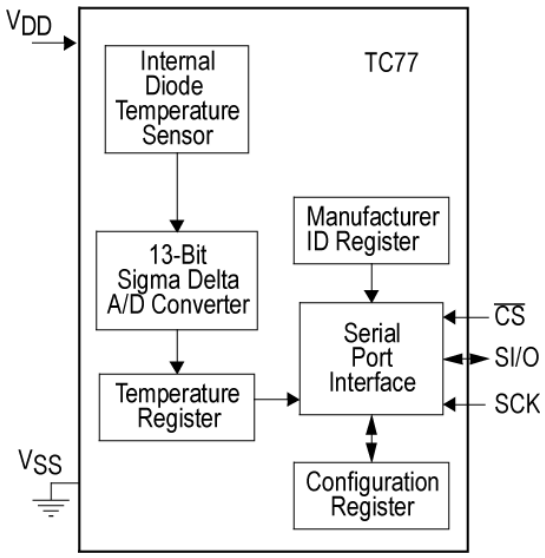


Figure 1) Schéma bloc du capteur de température TC77

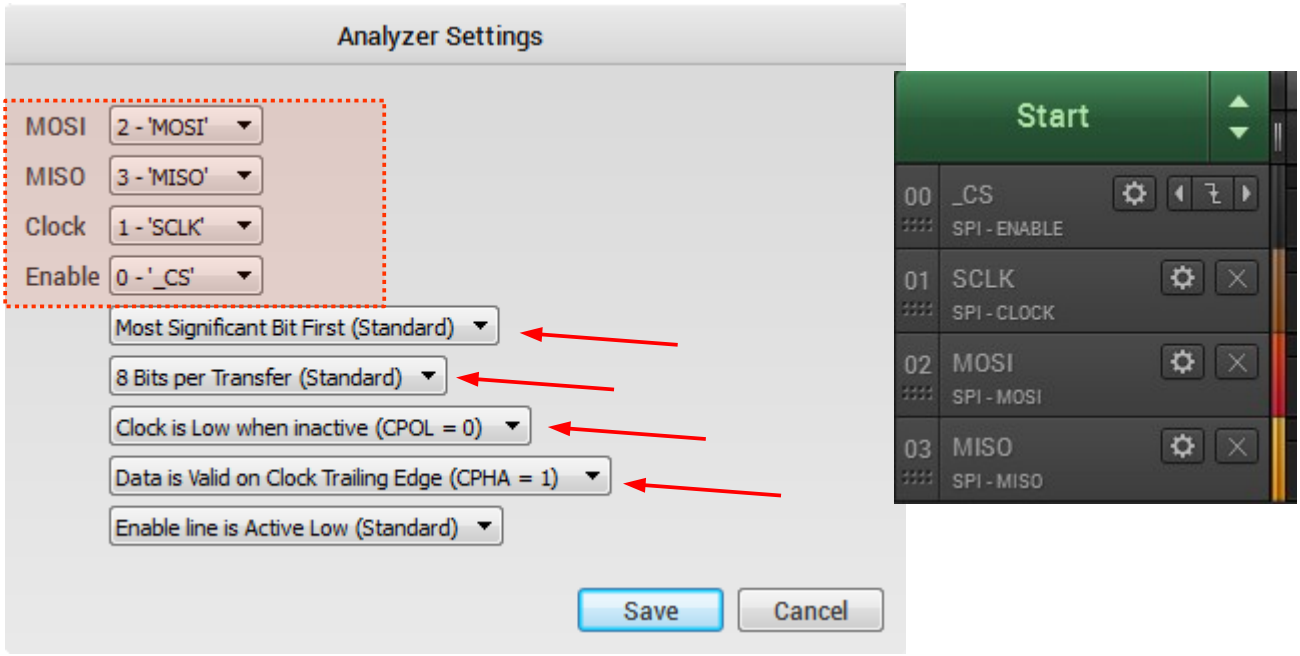
DC CHARACTERISTICS

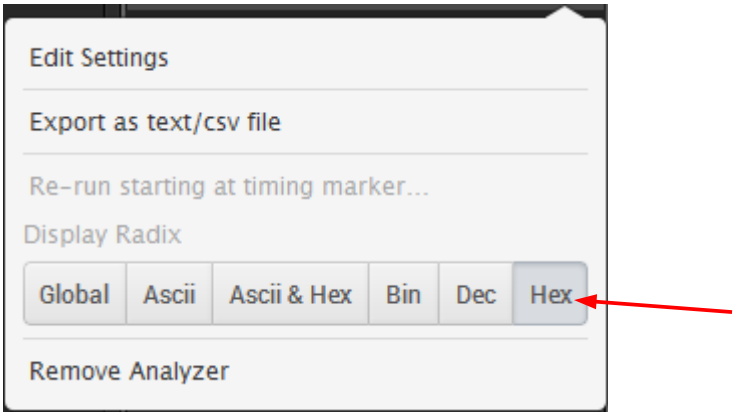
Electrical Specifications: Unless otherwise noted, all parameters apply at V_{DD} = 2.7V to 5.5V and T_A = -55°C to +125°C.

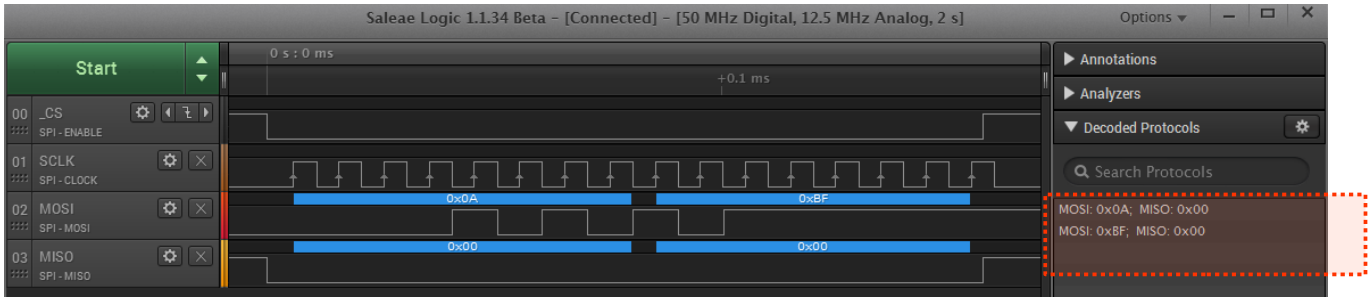
Parameters	Sym	Min	Typ	Max	Units	Conditions
Power Supply						
Operating Voltage Range	V_{DD}	2.7	—	5.5	V	**Note 1**
Operating Current	I_{DD}	—	250	400	µA	Continuous Temperature Conversion Mode
Power-On Reset Threshold	V_{POR}	1.2	1.6	2.2	V	V_{DD} falling or rising edge
Standby Supply Current	$I_{DD-STANDBY}$	—	0.1	1.0	µA	Shutdown Mode
Temperature to Bits Converter						
Resolution		—	13	—	Bits	ADC LSb = 0.0625°C/bit **(Note 4)**
Temperature Conversion Time	t_{CT}	—	300	400	ms	
Temperature Accuracy (**Note 1**)	T_{ERR}	-1.0 -2.0 -3.0	— — —	+1.0 +2.0 +3.0	°C	+25°C < T_A < +65°C -40°C < T_A < +85°C -55°C < T_A < +125°C TC77-3.3MXX: V_{DD} = 3.3V TC77-5.0MXX: V_{DD} = 5.0V

Figure 2) Vitesse de conversion du TC77

3.4 SALEA SPI settings







4 Travaux à effectuer

Ce chapitre résume les différents éléments à développer pour ce TP.

4.1 Préparation

Importer l'archive Micro-TP6.zip avec toutes les sources dans votre workspace.
Pour importer dans CCS : File → Import → CCS Projects → Select archive file.

Les paragraphes suivants donnent des indications sur les registres à configurer au niveau du microcontrôleur pour réaliser l'application

4.2 Partie 1: Configuration de l'eUSCI_B0

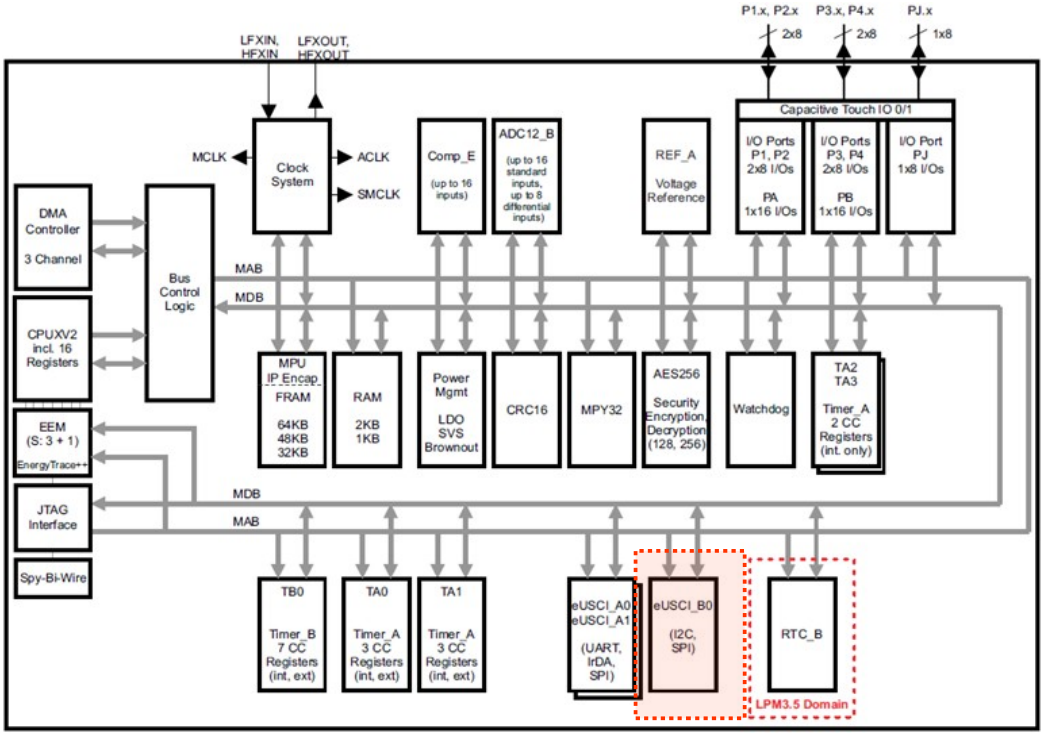


Figure 3) Diagramme de bloc MSP430FR5969

Comme le montre la figure 7, seule la ligne SOMI/MISO est utilisée pour les données. La ligne correspondante au signal est issue de la broche P1.7 du MSP430FR5969. L'horloge correspond au signal UCLK. Ici le capteur de température est considéré comme un Slave et le signal STE (ou CS) nécessaire est généré avec une sortie normale ce qui permet de faire des écritures/lectures de plusieurs bytes en même temps.

L'eUSCI_B0 est utilisé en mode SPI. L'interface SPI doit être configurée en tenant compte des spécifications du bus imposées par le capteur de température TC77 (fréquence d'horloge/flanc actif). Pour plus d'informations, veuillez consulter le chapitre 22.5 «eUSCI_B SPI Registers » du document «slau367f - MSP430FR58xx, MSP430FR59xx, MSP430FR68xx, and MSP430FR69xx Family User 's Guide.pdf» et le matériel de présentation SPI.

La figure suivante montre les possibilités de configurations (phase et polarité de l'horloge) de l'interface SPI du MSP430

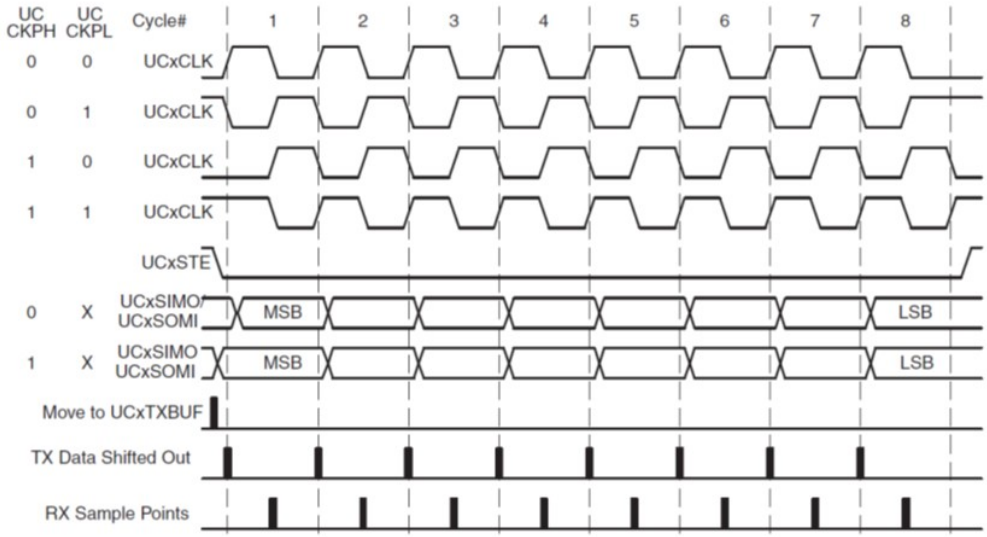


Figure 4) Chronogramme de l'interface eUSCI en mode SPI

La figure suivante montre les exigences du capteur TC77 :

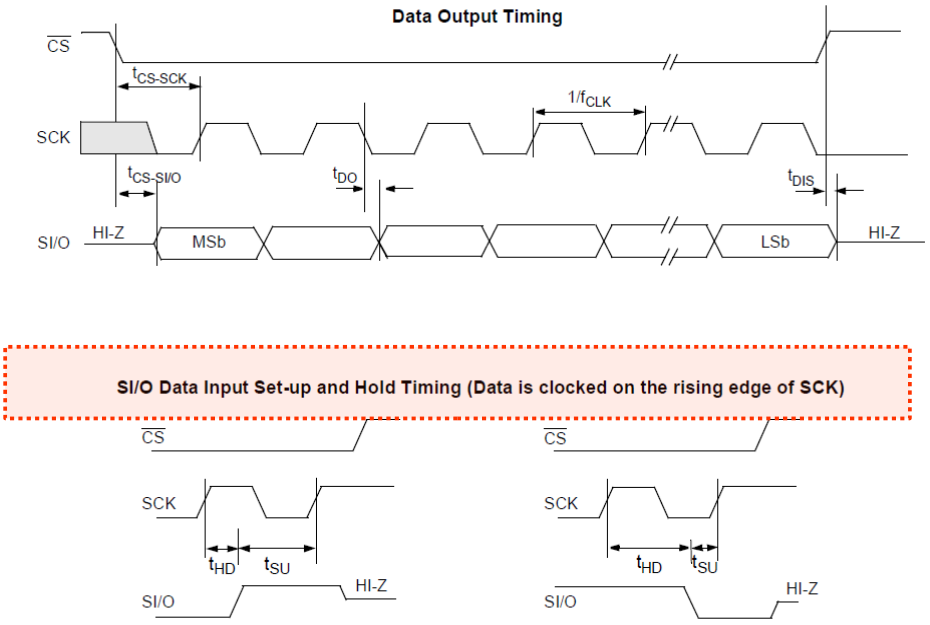


Figure 5) Flanc active du TC77

Après un rebuild, le compilateur indique où se trouve la partie manquante:

```
"../src/HAL/HAL_SPI.c", line xx: warning #36-D: #warn directive: 1) Implement SPI initialization as commented in header
```

Voir aussi le fichier header HAL_SPI.h qui contient des informations supplémentaires:

```
/// @brief  This method initializes the SPI to use MSP430FR5969 Launchpad
///         together with PICKit Demo board
///         e_USCI is initialized to operate as:
///         - SPI 8-bit Master @100 KHz
///         - Inactive Clock polarity is high
///         - 3-pin SPI mode (chip select is handled manually)
///         - source clock is SMCLK
void spi_init(void);
```

Attention toute la configuration doit se faire avec le flag SWRST à 1 (UCB0CTLW0 = UCSWRST).

Les registres à configurer sont:

- **Registre de contrôle de UCB0CTLW0**

  Les flags UCSWRST (SW Reset), UCSYNC(UART ou SPI mode), UCMODE (3SPI mode 3 ou 4 pins), UCMSB (MSB ou LSB first), UCCKPL (état inactif de l'horloge SCLK), UCMST (SPI master ou slave) doivent être configurés. Les autres bits sont à 0 par défaut et ne doivent pas être modifiés

- **Registres de contrôle du Baud rate de UCB0BRx**

  Ces deux registres sont utilisés pour fixer le Baud rate en divisant la fréquence de l'horloge de référence (SMCLK = 8 MHz dans notre cas). Le diviseur doit être configuré pour générer une fréquence d'horloge SCLK de 100kHz. Notez que $SCLK = \frac{f_{BRCLK}}{(UCBRx+1)}$.

Autres registres en relation avec l'interface SPI seulement si on l'utilisera avec les interruptions:

- **Registre de contrôle d'activation des interruptions UCB0IE**

  Le flag UCTXIE1 et le flag UXRXIE1 sont des flags de masquage d'interruption. Ils doivent les deux être forcés à 0 (défaut après reset) afin de s'assurer que les interruptions de l'eUSCI en mode SPI soient masquées.

- **Registre des flags d'interruptions UCB0IFG**

  Le flag UCRXIFG indique qu'un byte est disponible dans le buffer UCB0RXBUF.

4.3 Partie 2: Configuration GPIO pour SPI du module eUSCI_B0

Les signaux SPI du module eUSCI_B0 sont disponibles sur les ports 1 et 2.

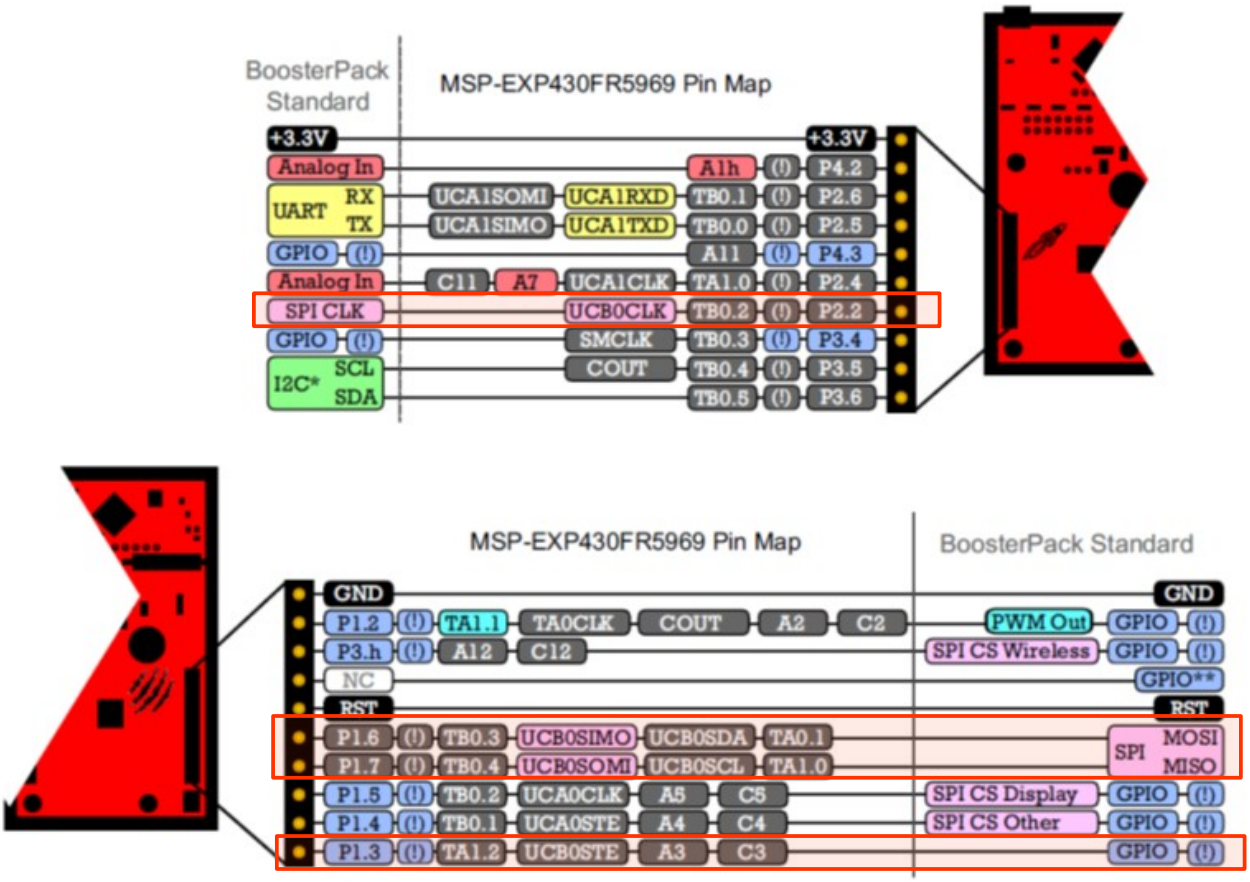


Figure 6) Signaux SPI sur le Launchpad MSP430FR5969

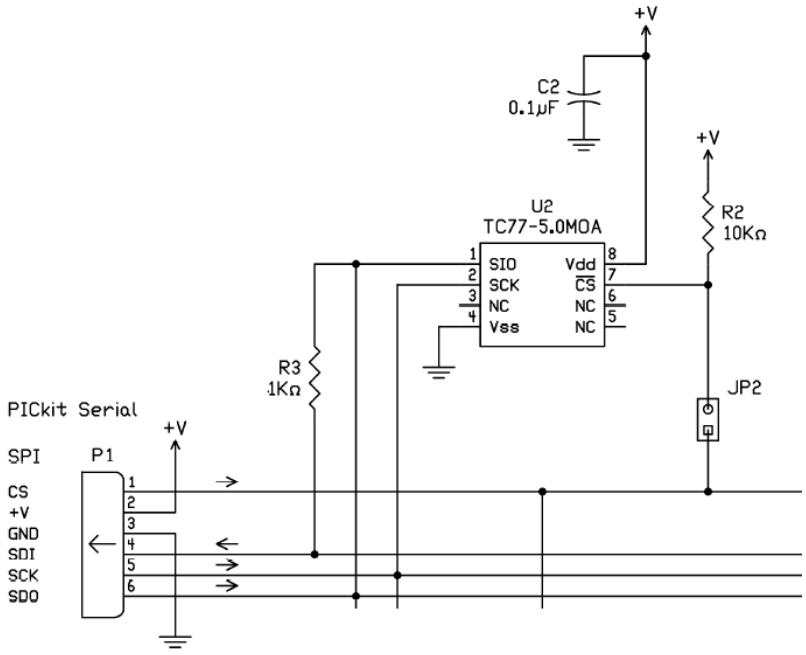


Figure 7) Signaux SPI sur le Microchip PICkit pour le TC77

Les broches sont assignées de la manière suivante :

- P1.6 MOSI (sortie pour le MSP430FR5969– pas utilisée pour le capteur de température) ;
- P1.7 MISO (entrée pour le MSP430FR5969) ;
- P2.2 SCLK (sortie pour le MSP430FR5969) ;
- P1.3 $\overline{CS}$ (sortie pour le MSP430FR5969 - utilisé comme GPIO) ;

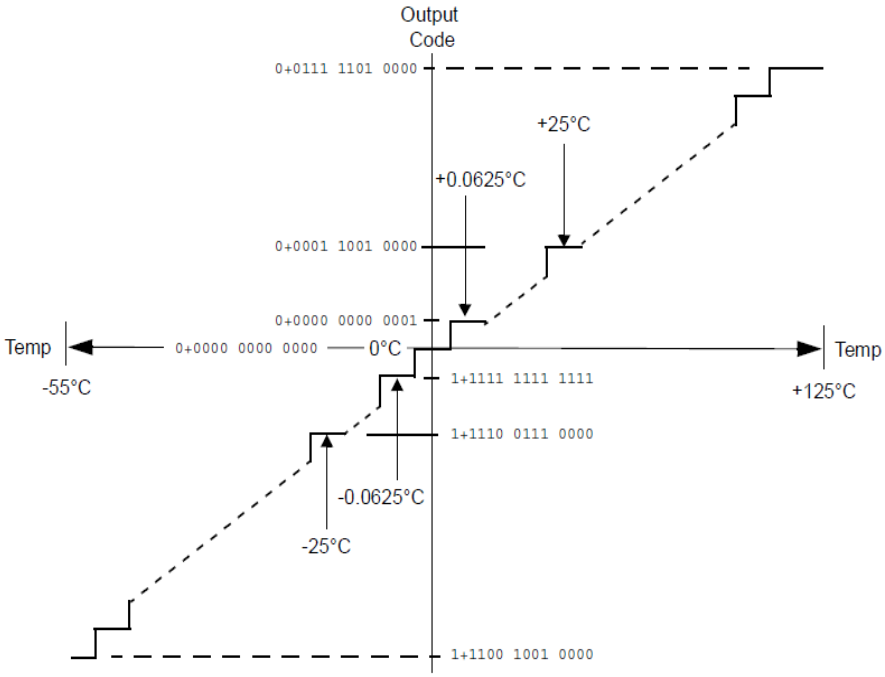
Les registres suivants doivent donc être configurés PxDIR , PxOUT, PxSEL .

Attention : seuls les bits correspondants aux broches utilisées doivent être modifiés.

Après un rebuild, le compilateur indique où se trouve la partie manquante:

```
"../src/HAL/HAL_MSP430.c", line xx: warning #36-D: #warn directive: 2) Implement SPI GPIO initialization
```

4.4 Partie 3: Implémentation de la lecture de température du TC77



La fonction de conversion est donnée par la figure suivante:

Il s'agit d'un code en complément à deux. La résolution (pas précision...) vaut 0.0625°C. Les deux bits de poids faibles ne doivent pas être pris en compte (bus en haute impédance). Le bit 2 est toujours à 1, excepté lors de la première conversion suivant une tension d'alimentation dépassant 1.6V (superviseur de tension intégré).

La valeur numérique utile est donc donnée sur les 13 bits de poids forts. Le tableau ci-dessous montre le codage de la température.

Température	Valeur numérique de sortie				
	Binaire				Hexadécimal
+125°C	0011	1110	1000	01zz	07D0
+25°C	0000	1100	1000	01zz	0190
+0.0625°C	0000	0000	0000	11zz	0001
0°C	0000	0000	0000	01zz	0000
-0.0625°C	1111	1111	1111	11zz	1FFF
-25°C	1111	0011	1000	01zz	1E70
-55°C	1110	0100	1000	01zz	1C90

Figure 8) Table de conversion

Le transfert des données par le bus SPI peut être effectué lorsque le signal $\overline{CS}$ est forcé au niveau bas, ainsi que le signal de l'horloge SCLK. Le bit de poids fort (MSB) est placé sur la ligne SI/0 au flanc descendant de $\overline{CS}$, puis chaque bit est placé sur le bus au flanc descendant du SCLK. Il faut donc au moins 13 périodes d'horloge pour une transmission complète d'une donnée. Les données doivent être capturées sur le flanc montant de l'horloge. Au flanc montant du signal $\overline{CS}$ la communication est interrompue. Elle recommencera lors du prochain flanc descendant de $\overline{CS}$.

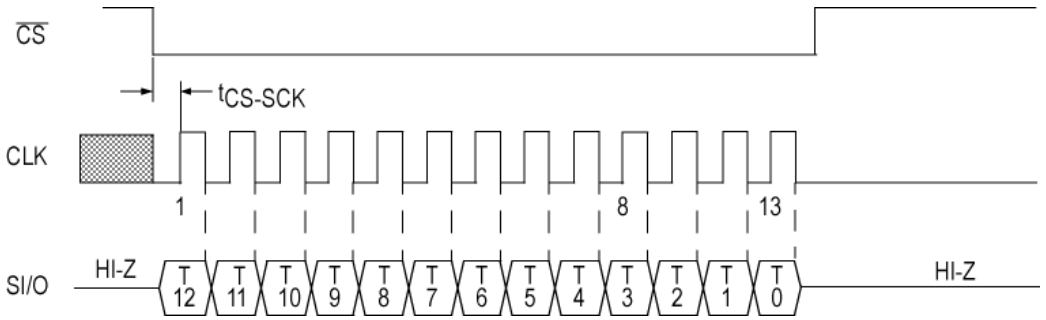


Figure 9) Mode lecture de la température

A l'enclenchement (power up) le capteur de température se trouve en mode de conversion continue de la température. Il est possible de changer de mode en plaçant le capteur en basse consommation (shutdown). Pour ce faire, il faut forcer le signal $\overline{CS}$ à 0, lire les 13 bits correspondant à une donnée (mesure de température), lire le bit suivant qui sera un 1 si la donnée est valide puis pour les deux dernières périodes de l'horloge, la ligne SI/0 se met en haute impédance. Les trois derniers bits ne font donc pas partie de la donnée.

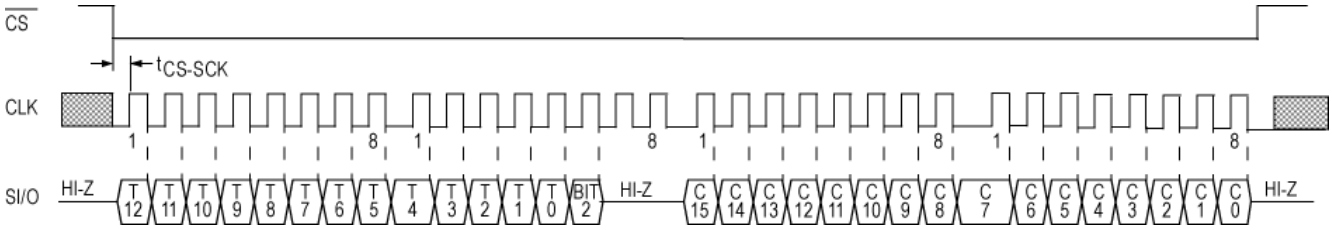


Figure 10) Mode lecture température puis écriture du registre de configuration

Sans changer le signal $\overline{CS}$ il est alors maintenant possible d'écrire dans le registre de contrôle. En écrivant 0x0000 le capteur travaillera en mode conversion continue, en écrivant 0xFFFF, le capteur se mettra en mode basse consommation. Une lecture en basse consommation permet de connaître l'ID du fabricant. Pour plus de détail voir le datasheet de Microchip.

Après un rebuild, le compilateur indique où se trouve la partie manquante:

```
"../src/TC77.c", line 43: warning #36-D: #warn directive: 3) Implement read of TC77 temperature sensor
```

Voir aussi le fichier header TC77.h qui contient des informations supplémentaires:

```
/// @brief  Reading TC77 temperature values using direct access
///         (without interrupts)
/// @param  -
/// @return 16 bits signed value (3 LSBs to be suppressed)
///         unit = 0.0625 °C/bit
///         returns INT_MAX if invalid
int16_t tc77_readTemp(void);
```

4.5 Partie 4: Implémentation de la conversion de température

Comme décrit dans le chapitre 3.3 *Capteur de température TC77* à la page 7, le capteur fournit une valeur signée (en complément à 2) avec 0.0625°C par bit. Comme on aimerait afficher la température avec une résolution de 0.1°C, il faut encore ajouter la conversion nécessaire en évitant d'utiliser des types à virgules flottantes.

Après un rebuild, le compilateur indique où se trouve la partie manquante:
`"../src/TP6_main.c", line xx: warning #36-D: #warn directive: 4) Implement necessary conversions for 0.1°C resolution`

```
int main(void)
{
  initChip();

  printf("\nTP6 - SPI with TC77 temperature sensor\n");

  while(1)
  {
    // measurement itself should be executed every 1s
    // note that TC77 temperature conversion takes up to 400ms...

    const int16_t cTempBin = tc77_readTemp(); // 13-bit binary format with 0.0625 °C/bit

    if (INT_MAX == cTempBin)
    {
      printf("Wait for temp available...\n");
    }
    else
    {
#warn 4) Implement necessary conversions for 0.1°C resolution
      const int16_t cTempConv = cTempBin; // in 0.1°C resolution


      printf("Temp = %3u.%u\n",cTempConv/10,abs(cTempConv%10));
    };

   __delay_cycles(8000000);

  }
}
```

4.6 Et à la fin...

1. Refaire un rebuild et Vérifier qu'il n'y a plus de warnings de compilateur.
2. Vérifier que le code correspond aux règles disponibles dans le document «C-Programming_Style_and_coding conventions.pdf»
3. Vérifier que tous les points sont traités – voir chapitre 2 *Résultat demandé* à la page 4
4. Exporter le projet pour le rendre avec le rapport: Export → Archive File → Donner un nom, (as zip)