MICROINFORMATIQUE

2EME SUPPLEMENT A LA NOTE D'APPLICATION 2

CONVERSION ANALOGIQUE-NUMERIQUE AVEC LE MODULE EZ430

1. Documents et matériel de référence

Le document de référence pour cette note d'application est :

1. ***Matériel : eZ430, potentiomètre.***
2. ***Slides : GPIO, Interruptions, Timers, Conversion Analogique-Numérique***
3. **MSP430-F2xx Family, User's Guide du F2012**, **Chapitre 20** (aussi accessible sur la page web des supports de cours).

2. Objectifs de cette note d'application

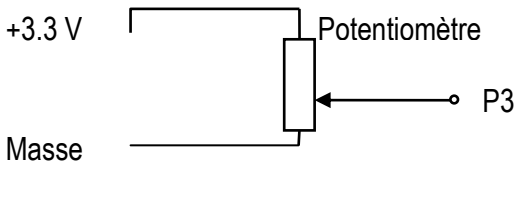
Le microcontrôleur MSP430F2012 inclut un convertisseur 10-bit avec 3 voies accessibles par des broches.

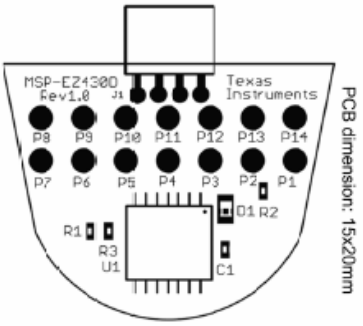
Il y a aussi un capteur interne de température analogique connecté à la voie A10.

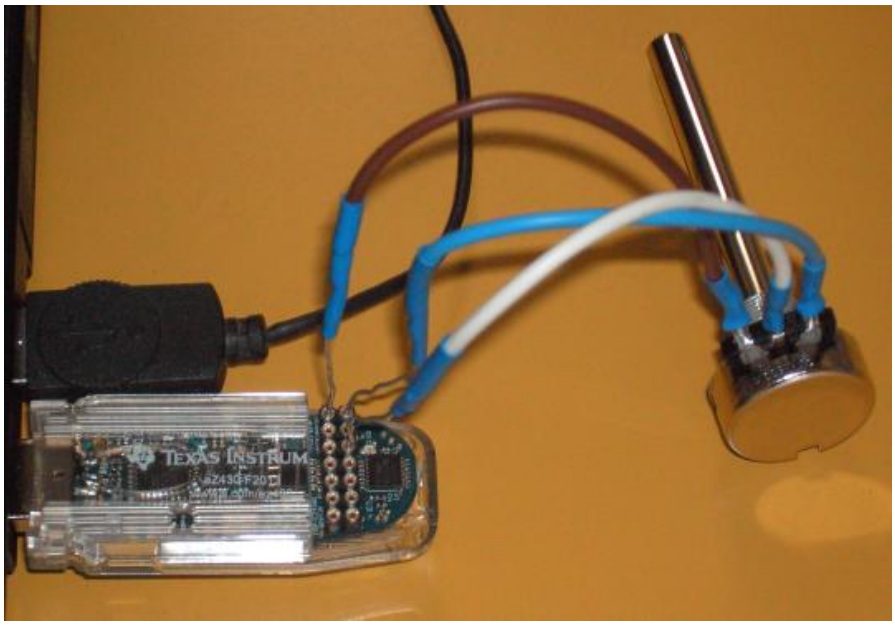
L'objectif de ces exercices est donc de programmer la lecture de tensions analogiques avec le module eZ430 et ainsi comprendre la gestion des registres de configuration et l'utilisation des interruptions spécifiques du convertisseur.

3. Exercices

1. Etudier le chapitre 20 (ADC10) du manuel de programmation (distribué avec cette note).
2. Insérer le potentiomètre entre les broches P1 (3,3 V), P14 (masse) et P3 (voie A1 de l'ADC) . En actionnant le potentiomètre on déterminera ainsi une tension variable entre 0 et 3,3 V sur la broche P3. Se référer aux schémas des pages 3 et 4 de la note d'application 2.







3. **Exercice de base** : créez un nouveau projet dans IAR et copier le programme suivant qui allume la LED interne du eZ430 (associée au port P1.0) quand la tension lue sur A1 dépasse un certain seuil (ici 1,65 V).

```
include "io430.h"
int main( void )
{
  // Stop watchdog timer to prevent time out reset
  WDTCTL = WDTPW + WDTHOLD;

  ADC10CTL0 = ADC10SHT_2 + ADC10ON +  REF2_5V + REFON;
  ADC10CTL1 = INCH_1;
  P1DIR |= 0x01;                            // P1.0 = output
  P1SEL |= 0x02;
    // ou équivalent   ADC10AE0 |= 0x02;  // P1.1 ADC option select

  P1OUT = 0;

  while(1)
  {
    ADC10CTL0 |= ENC + ADC10SC;    // Lancer échantillonage
    while (ADC10CTL1 & ADC10BUSY); // On attend que la conversion soit terminée
    if (ADC10MEM > 512) P1OUT = 0x01 ;
    else P1OUT = 0 ;
  }
}
```

Identifiez tous les registres et leurs composants et décrivez-en la fonction dans votre rapport.

4. On veut maintenant écrire le programme de manière que la boucle `while(1)` soit vide et que le traitement de valeur analogique lue advienne dans la routine d'interruption de l'ADC10. Pour cela on assignera le champ ADC10IE et activera ensuite les interruptions avec `__enable_interrupt()` – sans oublier l'include de `intrinsics.h`.

   Le premier lancement sera encore avant le `while(1)`. Ensuite tout se passe dans la routine :

```
#pragma vector=ADC10_VECTOR
__interrupt void ADC10_ISR (void)
{
    if (ADC10MEM > 500) P1OUT = 0x01;
    else P1OUT = 0;
    ADC10CTL0 |= ENC + ADC10SC;         // Nouvelle mesure
}
```

5. On demande de « afficher » la valeur analogique lue par une fréquence de clignotement variable : la fréquence sera zéro (LED allumée en continue) pour A1 = 0 et (environ) 20 Hz pour A1 = 3,3 V. Pour cela ajoutez un Timer_A, par exemple selon le modèle de l'exemple 10.2 de la note d'application. Le registre CCR0 sera ensuite modifié en fonction de la valeur du registre ADC10MEM.
   La routine d'interruption du timer lancée par `#pragma vector=TIMERA0_VECTOR` (voir toujours l'exemple 10.2) fera ensuite clignoter la LED à la fréquence voulue.

6. On veut ensuite dans un autre programme rendre plutôt l'intensité de la LED proportionnelle à la valeur analogique : éteinte pour A1 = 0, maximum pour A1 = 3,3 V. Pour cela on va créer un PWM à haute fréquence au moyen du Timer_A avec le SMCLK, dont le rapport cyclique (déterminé par le registre CCR1) sera proportionnel à ADC10MEM. Cette fois on pourra, par exemple, programmer la routine d'interruption du timer comme suit :

```
#pragma vector=TIMERA1_VECTOR
__interrupt void Timer_A(void)
{ unsigned int tar = TAR;
  if (TAIV)
  { if (tar <= CCR1)  P1OUT = 0x01;
    else P1OUT = 0x00;
  }
}
```

   Expliquez dans votre rapport le fonctionnement de cette routine. Autres variantes réalisant la même fonction sont bienvenues.

4. Rapport

Le rapport sera une mise au propre de vos propres notes prises durant ce travail : un simple log-book et enregistrement des divers programmes réalisés et testés.
Il inclura pour chaque tache effectuée

- Titre de la tache.
- Copier-coller des *parties significatives* des listings produits.
- Les réponses à toutes les questions posées.
- Remarques, résultats, conclusions et suggestions applicables pour chaque cas.