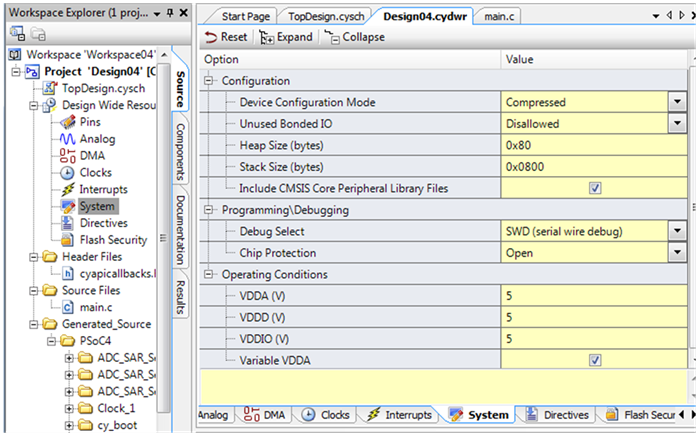
**Objectif : Faire varier la luminosité d’une LED, en fonction de la rotation d’un potentiomètre**

# Préparation de la carte PSoC



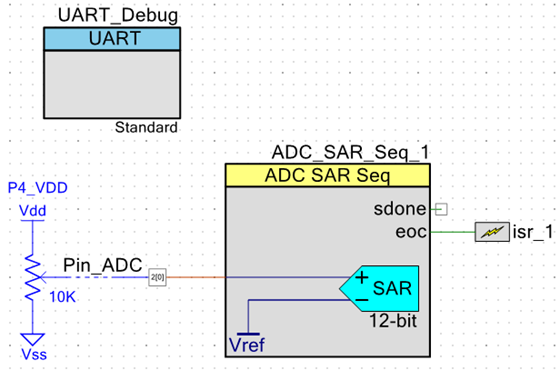
Hors-alimentation de la carte, mettre le cavalier d’alimentation repéré J9 sur 5 V.

Vous devez également renseigner la configuration en 5 V dans l'environnement PSoC Creator. Dans l'onglet avec l'extension .cydwr, sélectionnez l'onglet *System* en bas de la fenêtre. Renseignez les champs avec la nouvelle configuration (au besoin, fermer et rouvrir PSoC Creator pour prendre en compte les changements).



# *Design* partie 1

* Connectez le potentiomètre entre les connecteurs GND (masse) et P4\_VDD. Le curseur sera relié au connecteur **2.0** de la carte.
* Le *design* à concevoir dans un nouveau projet est le suivant (voir le détail pages suivantes) :



Liaison série standard UART

Convertisseur Analogique-numérique par Approximations Successives 12 bits

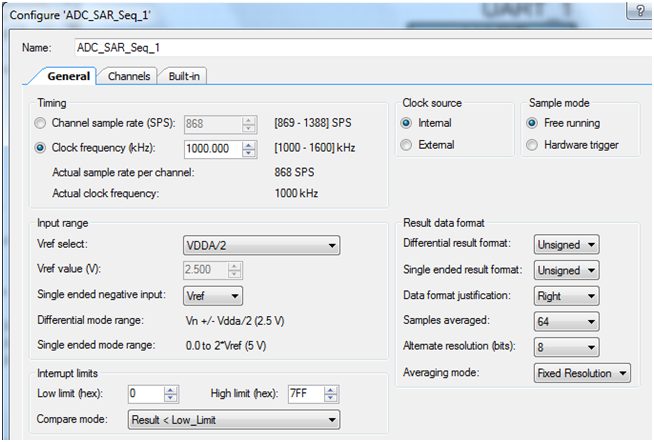
Interruption

eoc = *End Of Conversion*

Entrée analogique dirigée vers 2[0]

L’entrée analogique sera lue et la conversion analogique-numérique effectuée en continu. A chaque fin de conversion, un signal **eoc** *(End of Conversion*) déclenchera une interruption. Dans le code interruption, la valeur issue de la conversion est dans un premier temps récupérée pour être envoyée sur un terminal série pour consultation.

# Configuration de la conversion analogique-numérique



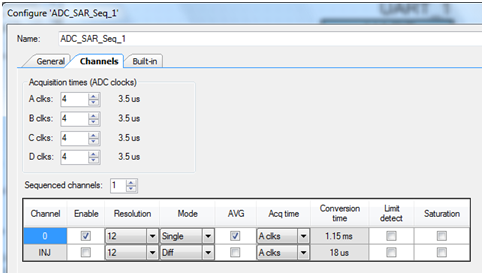
Fréquence d’horloge : 1000 kHz, soit 868 SPS (*samples per Second*)

*Mode Single Ended* entre 0 et 5 V

Echantillonnage en *free running*

Moyenne sur 64 échantillons

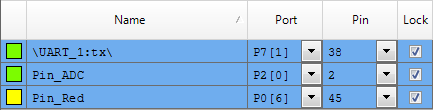
Format du résultat : *unsigned*



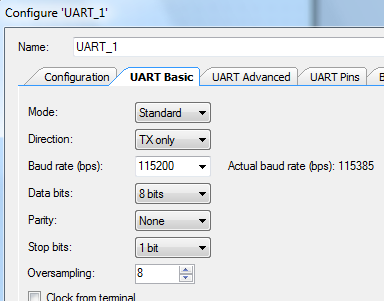
Résolution 12 bits

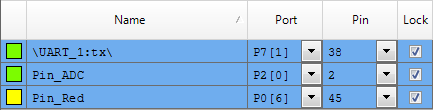
Temps de conversion : 1,15 ms

# Configuration Entrée analogique



# Configuration Liaison série UART





# Code (dans le fichier principal *main.c*)

|  |
| --- |
| #include <project.h>  #include <stdio.h>  volatile uint8 flagEOC = 0;  volatile int16 result = 0;  CY\_ISR(ADCEOC) { // code d'interruption  /\* result = valeur issue de la conversion analogique-numérique  result en 0 et 4095  on met un drapeau (flag) à 1 pour signaler  qu'une conversion est terminée \*/  result = ADC\_SAR\_Seq\_1\_GetResult16(0);  flagEOC = 1;  }  int main() {  CyGlobalIntEnable; /\* Enable global interrupts \*/  isr\_1\_StartEx(ADCEOC);    ADC\_SAR\_Seq\_1\_Start();  ADC\_SAR\_Seq\_1\_Enable();  ADC\_SAR\_Seq\_1\_StartConvert();  UART\_Debug\_Start();  for (;;) { // boucle principale  if (flagEOC) { // si un résultat est disponible  flagEOC = 0;  char8 s[12];  sprintf(s,"%u\r\n",result);  UART\_Debug\_UartPutString(s); // on l'envoie vers le terminal Série  }  }  } |

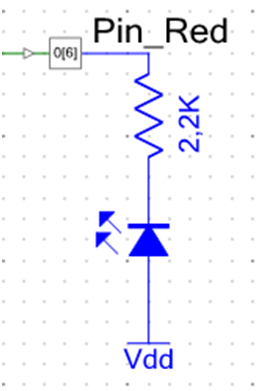
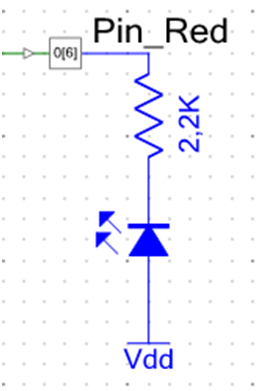
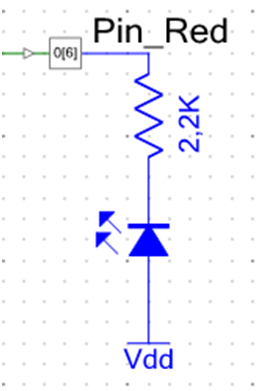
# Premiers tests

Après configuration, compilation et transfert du code dans la carte, ouvrir un terminal Série (*RealTerm*).

En agissant sur le potentiomètre, les valeurs devraient évoluer entre 0 et 4095.

# Deuxième partie : faire varier la luminosité de la LED rouge

En jouant sur le rapport cyclique d’un signal à haute fréquence modulé en largeur d’impulsion (*Pulse Width Modulation*), on peut faire varier la luminosité d’une LED.



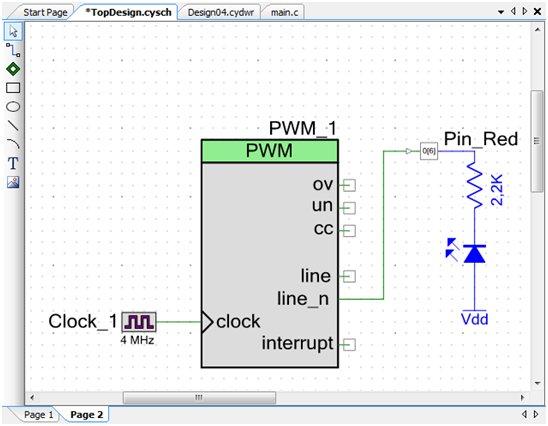
LED fortement lumineuse

LED faiblement lumineuse

LED avec luminosité moyenne

# Design partie 2

Compléter **dans un deuxième onglet** avec le *design* suivant :



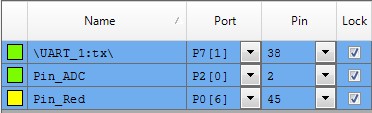
Générateur de signaux modulés en largeur d’impulsion (PWM).

*Design* dans un 2e onglet

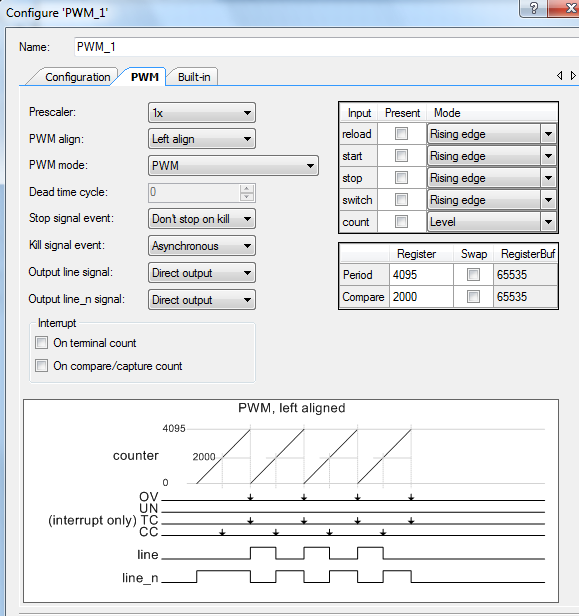
Sortie numérique dirigée vers 0[6] (connectée à la LED rouge).

Horloge à 4  MHz

# Configuration de la sortie numérique



# Configuration du générateur PWM



Un compteur (*counter* sur la figure ci-dessus) s’incrémente entre 0 et 4095 (valeur du paramètre *period*). Sur débordement du compteur, il repart à zéro. La vitesse d’incrémentation du compteur dépend de la fréquence de l’horloge.

Lorsque le compteur atteint 2000 (valeur du paramètre *compare*), le signal ***line\_n*** bascule. Il bascule à nouveau lorsque le compteur atteint 4095.

**Constatez l’évolution du signal de sortie *line\_n* lorsque vous faites varier la valeur du paramètre *Compare*.**

# Le code

On repart du code précédent dans le fichier *main.c* (modifications **en gras**).

|  |
| --- |
| #include <project.h>  #include <stdio.h>  volatile uint8 flagEOC = 0;  volatile int16 result = 0;  CY\_ISR(ADCEOC) { // code d'interruption  /\* result = valeur issue de la conversion analogique-numérique  result en 0 et 4095  on met un drapeau (flag) à 1 pour signaler  qu'une conversion est terminée \*/  result = ADC\_SAR\_Seq\_1\_GetResult16(0);  flagEOC = 1;  }  int main() {  CyGlobalIntEnable; /\* Enable global interrupts \*/  isr\_1\_StartEx(ADCEOC);    ADC\_SAR\_Seq\_1\_Start();  ADC\_SAR\_Seq\_1\_Enable();  ADC\_SAR\_Seq\_1\_StartConvert();  UART\_Debug\_Start();    **PWM\_1\_Start(); // Démarrage du générateur PWM**  for (;;) { // boucle principale  if (flagEOC) { // si un résultat est disponible  flagEOC = 0;  char8 s[12];  sprintf(s,"%u\r\n",result);  UART\_Debug\_UartPutString(s); // on l'envoie vers le terminal Série  **PWM\_1\_WriteCompare(result); // modification du paramètre Compare du PWM**  }  }  } |

La ligne **PWM\_1\_WriteCompare(result);** permet de modifier le paramètre *Compare* du générateur PWM, en fonction de la rotation du potentiomètre.

# Tests

Mettre en œuvre le programme et transférez-le dans la carte. L’image de la rotation du potentiomètre est toujours visible dans le terminal Série de *SerialTerm*.

Constatez que l’intensité lumineuse de la LED rouge varie en fonction de la rotation du potentiomètre.

# Exercice supplémentaire

Connectez l’autre sortie **line** du générateur PWM vers une nouvelle sortie numérique. Cette sortie numérique sera dirigée vers la broche **2[6]** (qui est reliée à la LED verte de la LED multicolore). Le programme du fichier *main.c* précédent est conservé tel quel.

Générez à nouveau le projet et effectuez le transfert vers la carte.

Testez le nouveau programme en actionnant le potentiomètre. Que se passe-t-il ? Expliquez pourquoi.