



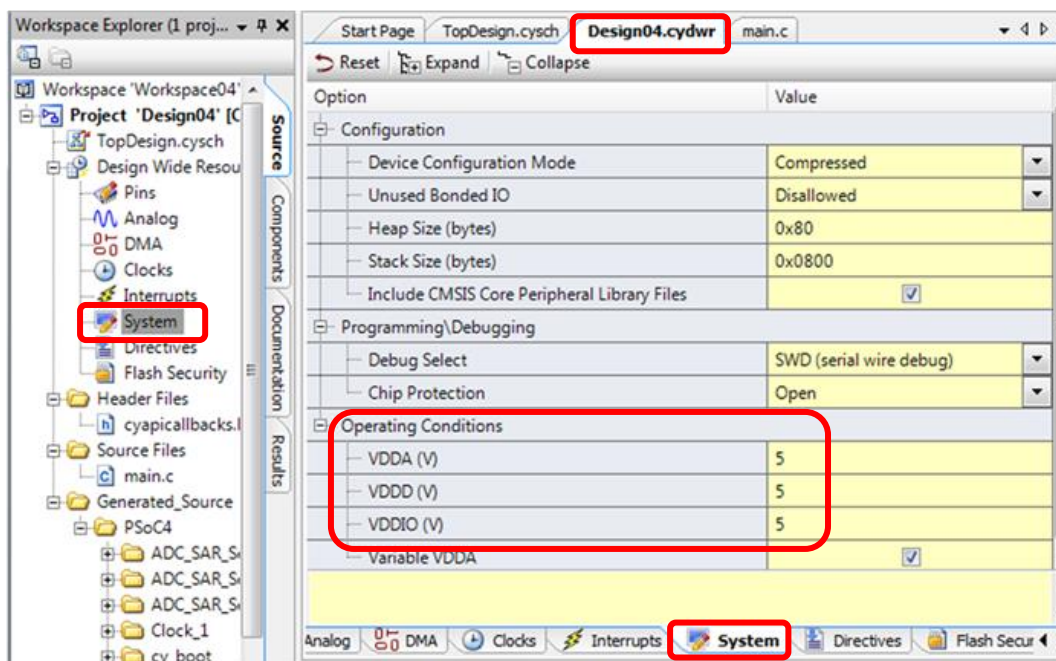
Objectif : Faire varier la luminosité d'une LED, en fonction de la rotation d'un potentiomètre

Préparation de la carte PSoC



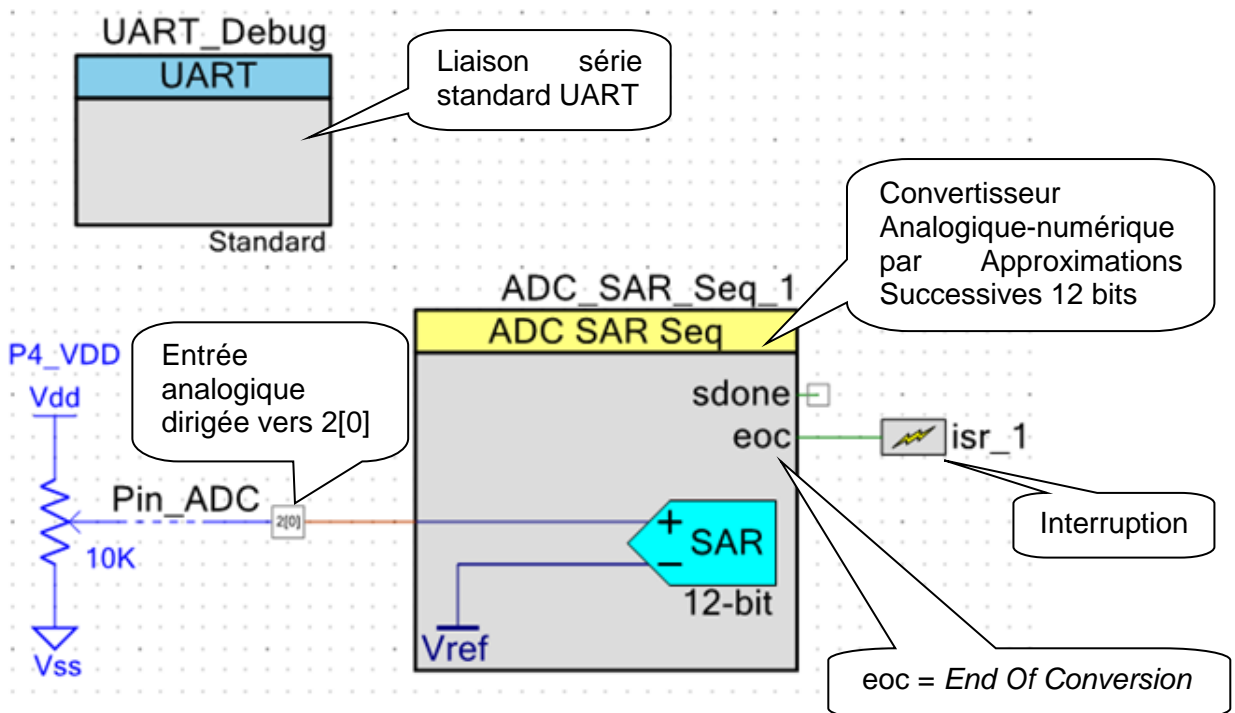
Hors-alimentation de la carte, mettre le cavalier d'alimentation repéré J9 sur 5 V.

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) :



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

Configure 'ADC_SAR_Seq_1'

Name: ADC_SAR_Seq_1

General Channels Built-in

Timing

☐ Channel sample rate (SPS): 868 [869 - 1388] SPS

☒ Clock frequency (kHz): 1000.000 [1000 - 1600] kHz

Actual sample rate per channel: 868 SPS

Actual clock frequency: 1000 kHz

Input range

Vref select: VDDA/2

Vref value (V): 2.500

Single ended negative input: Vref

Differential mode range: Vn +/- Vdda/2 (2.5 V)

Single ended mode range: 0.0 to 2*Vref (5 V)

Interrupt limits

Low limit (hex): 0 High limit (hex): 7FF

Compare mode: Result < Low_Limit

Clock source

☒ Internal

☐ External

Sample mode

☒ Free running

☐ Hardware trigger

Result data format

Differential result format: Unsigned

Single ended result format: Unsigned

Data format justification: Right

Samples averaged: 64

Alternate resolution (bits): 8

Averaging mode: Fixed Resolution

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*

Configure 'ADC_SAR_Seq_1'

Name: ADC_SAR_Seq_1

General **Channels** Built-in

Acquisition times (ADC clocks)

A clks: 4 3.5 us

B clks: 4 3.5 us

C clks: 4 3.5 us

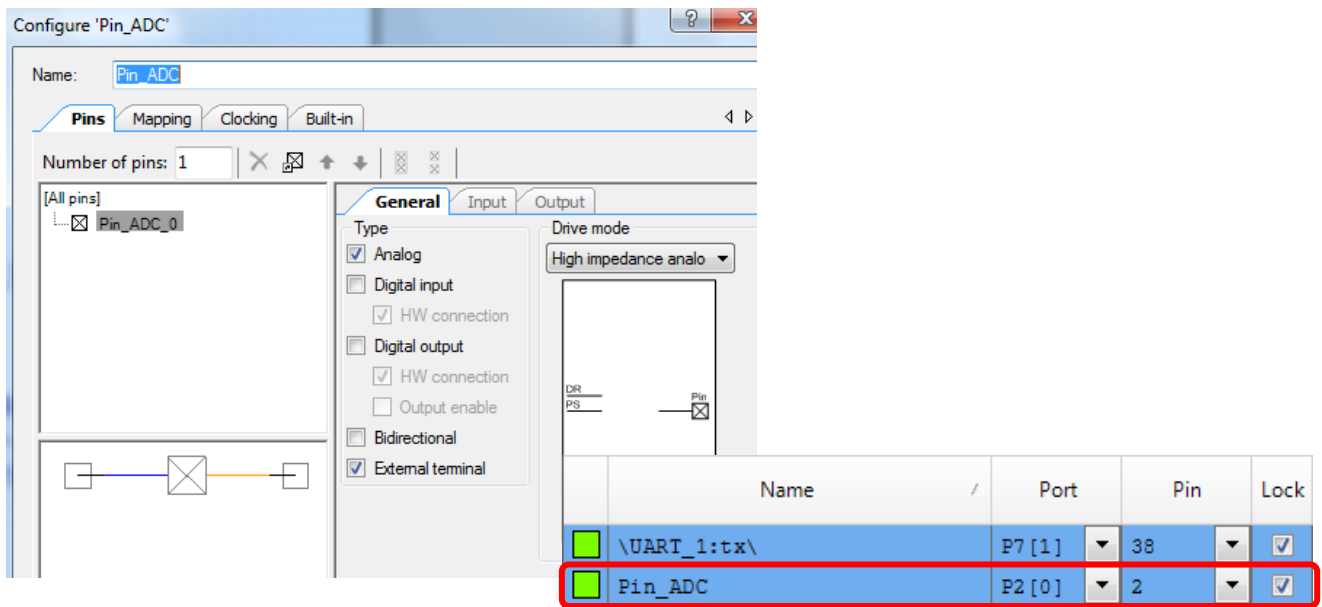
D clks: 4 3.5 us

Sequenced channels: 1

Channel	Enable	Resolution	Mode	AVG	Acq time	Conversion time	Limit detect	Saturation
0	<input checked="" type="checkbox"/>	12	Single	<input checked="" type="checkbox"/>	A clks	1.15 ms	<input type="checkbox"/>	<input type="checkbox"/>
INJ	<input type="checkbox"/>	12	Diff	<input type="checkbox"/>	A clks	18 us	<input type="checkbox"/>	<input type="checkbox"/>

Résolution 12 bits
 Temps de conversion : 1,15 ms

Configuration Entrée analogique



Configure 'Pin_ADC'

Name: Pin_ADC

Number of pins: 1

[All pins]

- Pin_ADC_0

General

Type

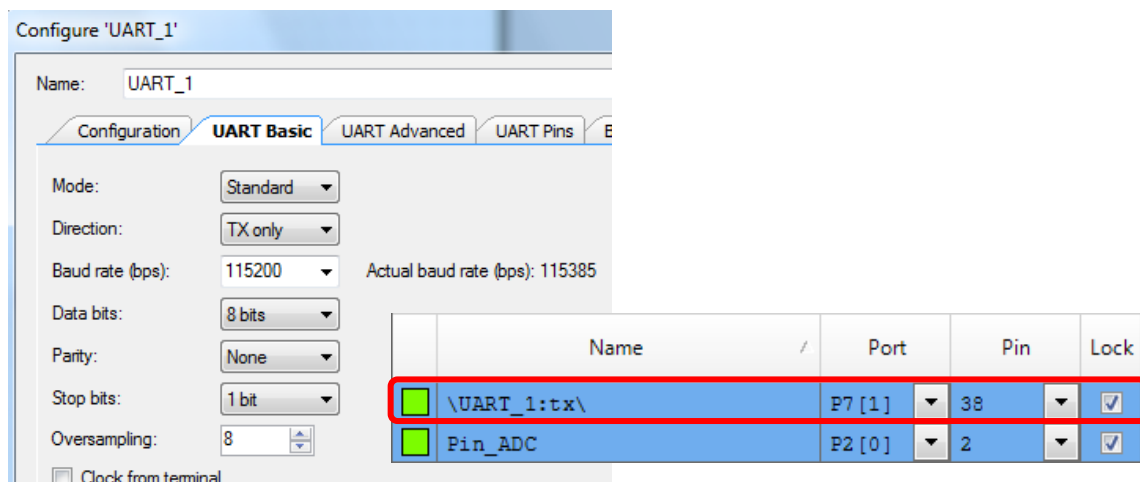
- ☒ Analog
- ☐ Digital input
 - ☒ HW connection
- ☐ Digital output
 - ☒ HW connection
 - ☐ Output enable
- ☐ Bidirectional
- ☒ External terminal

Drive mode

High impedance analog

	Name	Port	Pin	Lock
<input checked="" type="checkbox"/>	\UART_1:tx\	P7 [1]	38	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	Pin_ADC	P2 [0]	2	<input checked="" type="checkbox"/>

Configuration Liaison série UART



Configure 'UART_1'

Name: UART_1

Configuration

UART Basic

Mode: Standard

Direction: TX only

Baud rate (bps): 115200 Actual baud rate (bps): 115385

Data bits: 8 bits

Parity: None

Stop bits: 1 bit

Oversampling: 8

☐ Clock from terminal

	Name	Port	Pin	Lock
<input checked="" type="checkbox"/>	\UART_1:tx\	P7 [1]	38	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	Pin_ADC	P2 [0]	2	<input checked="" type="checkbox"/>

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();

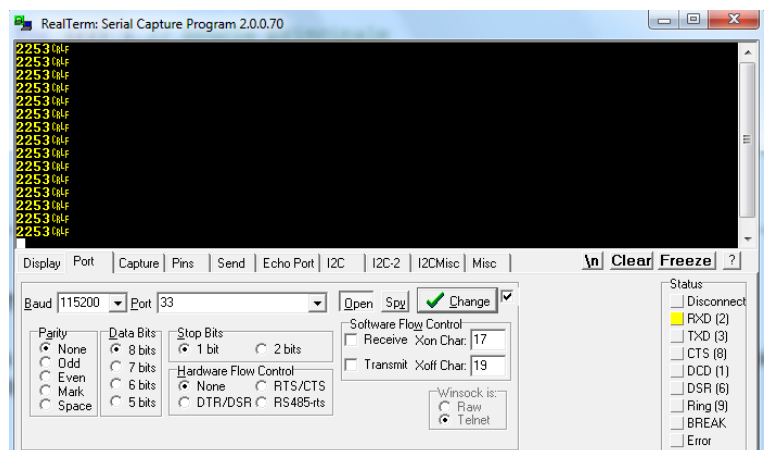
    UART_Debug_Start();

    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

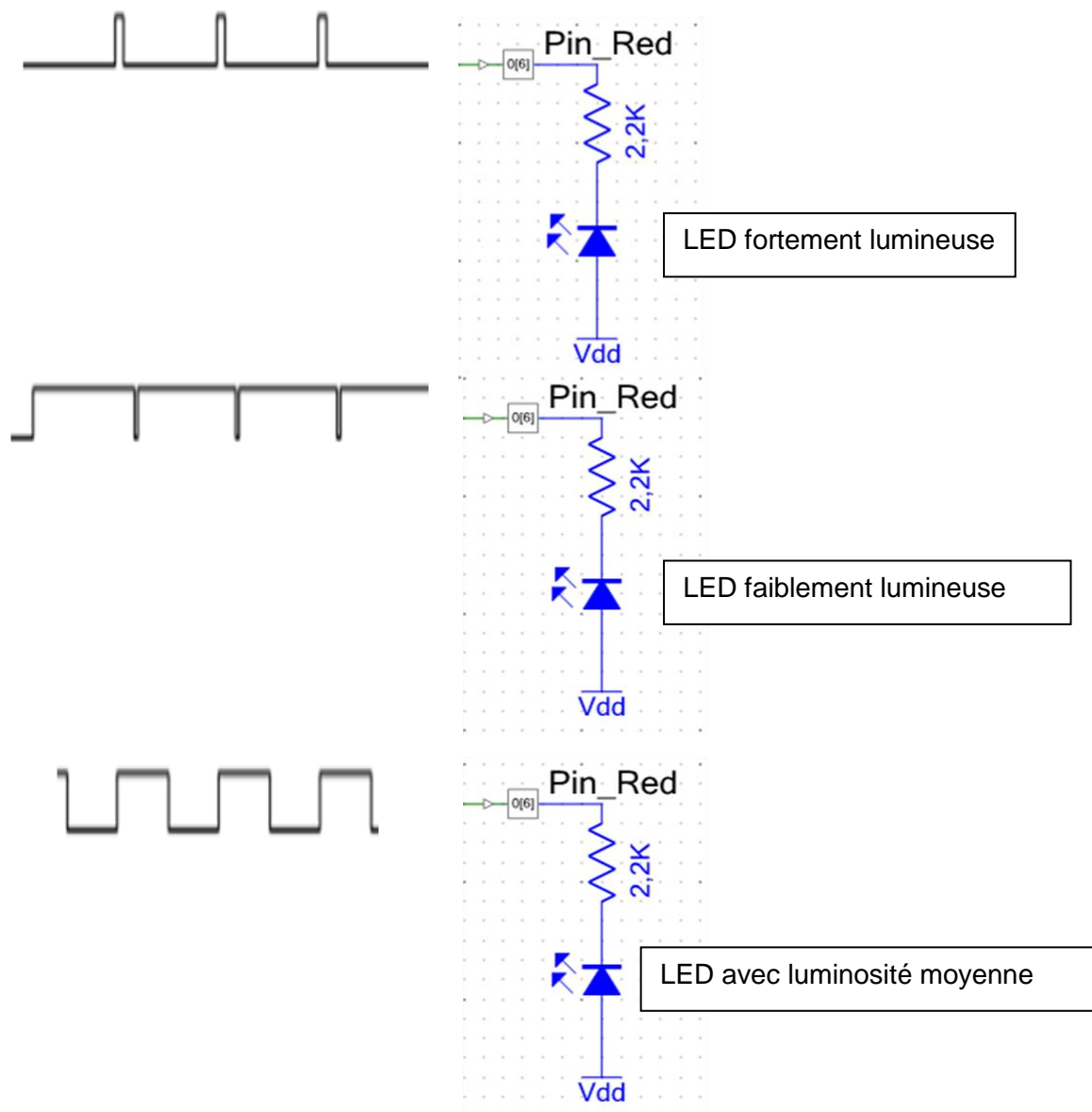
Après configuration, compilation et transfert du code dans la carte, ouvrir un terminal Série (*RealTerm*).

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



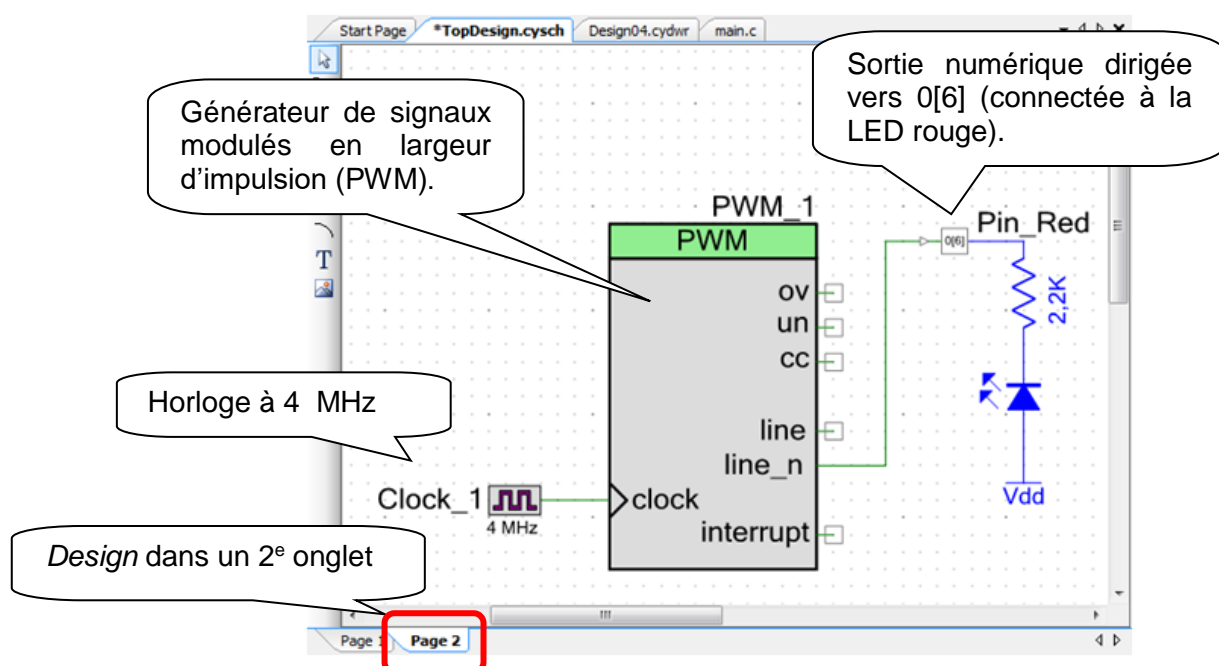
Deuxième partie : faire varier la luminosité de la LED rouge

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.



Design partie 2

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



Configuration de la sortie numérique

Configure 'Pin_Red'

Name: Pin_Red

Pins Mapping Cloning Built-in

Number of pins: 1

[All pins] Pin_Red_0

General Input Output

Type

☐ Analog

☐ Digital input

☒ HW connection

☒ Digital output

☒ HW connection

☐ Output enable

☐ Bidirectional

☒ External terminal

Drive mode

Strong drive

Name	Port	Pin	Lock
\UART_1:tx\	P7 [1]	38	<input checked="" type="checkbox"/>
Pin_ADC	P2 [0]	2	<input checked="" type="checkbox"/>
Pin_Red	P0 [6]	45	<input checked="" type="checkbox"/>

Configuration du générateur PWM

Configure 'PWM_1'

Name: PWM_1

Configuration **PWM** Built-in

Prescaler: 1x

PWM align: Left align

PWM mode: PWM

Dead time cycle: 0

Stop signal event: Don't stop on kill

Kill signal event: Asynchronous

Output line signal: Direct output

Output line_n signal: Direct output

Interrupt

☐ On terminal count

☐ On compare/capture count

Input	Present	Mode
reload	<input type="checkbox"/>	Rising edge
start	<input type="checkbox"/>	Rising edge
stop	<input type="checkbox"/>	Rising edge
switch	<input type="checkbox"/>	Rising edge
count	<input type="checkbox"/>	Level

	Register	Swap	RegisterBuf
Period	4095	<input type="checkbox"/>	65535
Compare	2000	<input type="checkbox"/>	65535

PWM, left aligned

counter

4095

2000

0

OV

UN

(interrupt only) TC

CC

line

line_n

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.