

GBM2100

Projet d'instrumentation biomédicale



Laboratoire 3 – eINK display, PWM et ADC

Préparé par :

Christophe Cloutier-Tremblay

Mise à jour le 23 février 2021

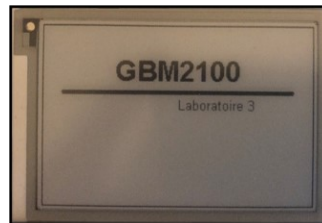
Introduction

Ce troisième laboratoire vous permettra d'appliquer les concepts vus au dernier cours. Vous allez générer un PWM et l'acquiesionner à l'aide d'un ADC. Vous utiliserez également l'écran de votre plaquette pour afficher l'allure de la courbe échantillonnée et afficher la moyenne et le cycle efficace (duty cycle).

Un projet vous est fourni pour ce laboratoire. Aller le chercher sur le site Moodle du cours et importer le dans votre IDE. Ouvrez le fichier **Laboratoire3.cywrk**. Accepter tout, ne cocher pas les fichiers avant de « replace ». Si vous avez l'erreur : «missing GUI.h». Référez-vous à la dernière section de ce laboratoire.

Pour ce laboratoire, nous utiliserons l'autre cœur, le cortex M4. L'écriture du programme se fera donc dans le fichier main_cm4.c. **Attention, le cœur cmop contient le programme de votre laboratoire 2 !** Assurez-vous de le reprogrammer avec le nouveau projet. Il faut donc programmer une seule fois le cmop et ensuite le cm4. Pour les programmations subséquentes, il n'est pas nécessaire de reprogrammer le cmop.

Avant de commencer à modifier le projet, téléverser le code sur votre microcontrôleur pour vous assurer que votre projet n'a pas d'erreur. Vous devriez voir apparaître GBM2100, Laboratoire 3 sur votre écran.



Évaluation

ÉLÉMENT	PONDÉRATION
QUESTIONS	5
CODE PARTIE 1	4
CODE PARTIE 2	2
CODE PARTIE 3	9
TOTAL	20

Vous pouvez répondre aux questions dans un fichier distinct ou écrire au début de votre main_cm4.c sous forme de commentaire. Vous devrez remettre votre projet PsOC Creator, pour se faire :

File -> Create Workspce Bundle. La source doit être le Workspace, la destination de sauvegarde est laissée à votre discrétion, cocher Compress Archive (zip) et le type doit être Minimal. Cliquez sur Archive.

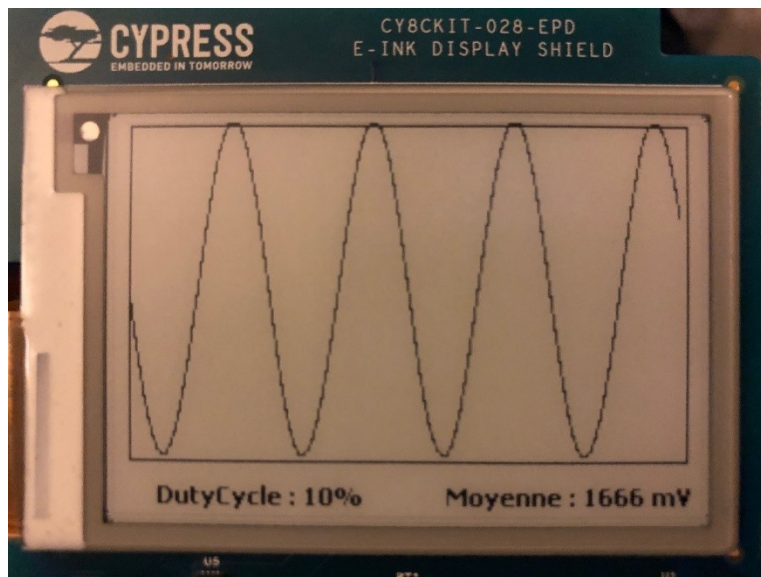
Partie 1 – eINK display

En premier lieu, vous devrez vous familiariser avec l’affichage graphique à l’aide de la librairie du module CY8CKIT-028-EPD. Les fonctions disponibles se retrouvent dans le fichier GUI.h (PSoC6\pdl\middleware\emWin\code\include\GUI.h). La résolution de l’écran est de 264 x 176 pixels.

Code 1.1 (2pts) : Vous devez tracer un graphique à partir d’un vecteur de 125 éléments. Vous devez donc écrire une fonction qui prend en paramètre un pointeur vers un vecteur de 125 éléments variant de -1 à 1 et qui affiche le graphique sur l’écran. La figure suivante montre l’affichage d’un sinus dont le vecteur est inclus à la ligne 57. Votre graphique devra prendre **250x150** de l’écran. Le prototype de la fonction est le suivant : `void drawGraph(*vector150elements)`.

```
55 |
56 | //vecteur sinus de 125 élément entre -1 et 1
57 | float vector[]={0.099833,0.29552,0.47943,0.64422,0.78333,0.89121,0.96356,0.99749,0.99
```

Code 1.2 (2 pts) : Écrivez une autre fonction qui permettra d’actualiser 2 paramètres au bas de l’écran, le Duty Cycle et la moyenne du signal. Le prototype de cette fonction est le suivant : `void updateParameters(param1, param2)`.



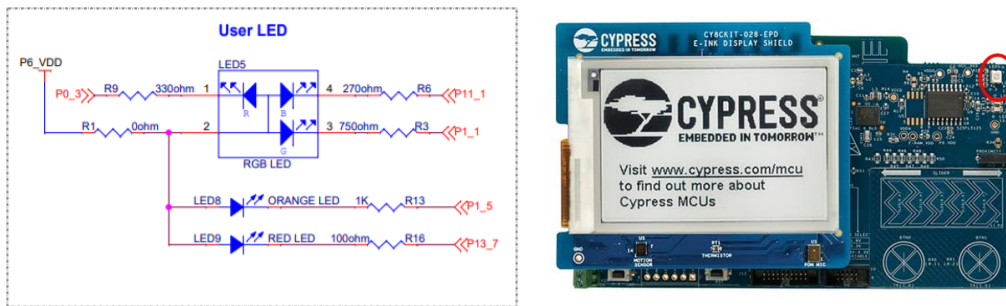
Question 1.1 (1pt) : En vous référant à la datasheet, cet écran est-il approprié pour un affichage en temps réel à une fréquence d’échantillonnage de 10 Hz. Appuyer votre réponse par une/des valeurs numériques.

Datasheet : https://www.pervasivedisplays.com/wp-content/uploads/2019/06/1P138-00_03_E2271CS021_20180418.pdf

Partie 2 – PWM

Dans le même projet, vous devrez générer un train de pulse qui permettra de contrôler l'intensité de la LED verte de votre plaquette. Trouvez la pin associée dans le schéma disponible ici-bas et ici : <https://www.cypress.com/file/420836/download>.

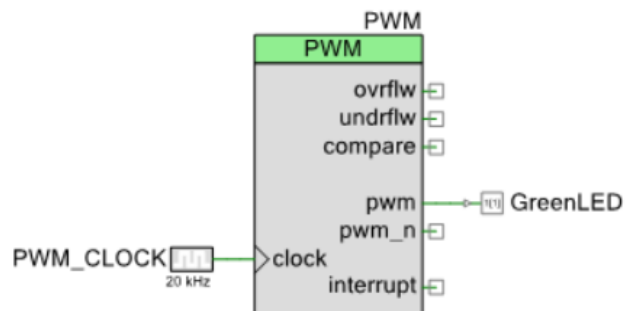
Question 2.1 (1pt) : À quel moment la LED devrait-elle s'allumer ? Lorsque la pin est HIGH ou LOW ? Quel mode devriez-vous configurer la pin ?



Code 2.1 (2 pts) : Insérez un module PWM, une horloge et une pin dans le schéma pour générer un train de pulse sur la LED verte. Dans le module PWM, modifiez les paramètres « period » et « compare » et regardez l'effet sur la LED. Ensuite, modifier la fréquence de l'horloge et regarder l'effet sur la LED. Utilisez les paramètres suivants comme point de départ :

Clock	20 kHz
Period	1000 counts
Compare	500 counts

Votre schéma devrait ressembler à la figure suivante.

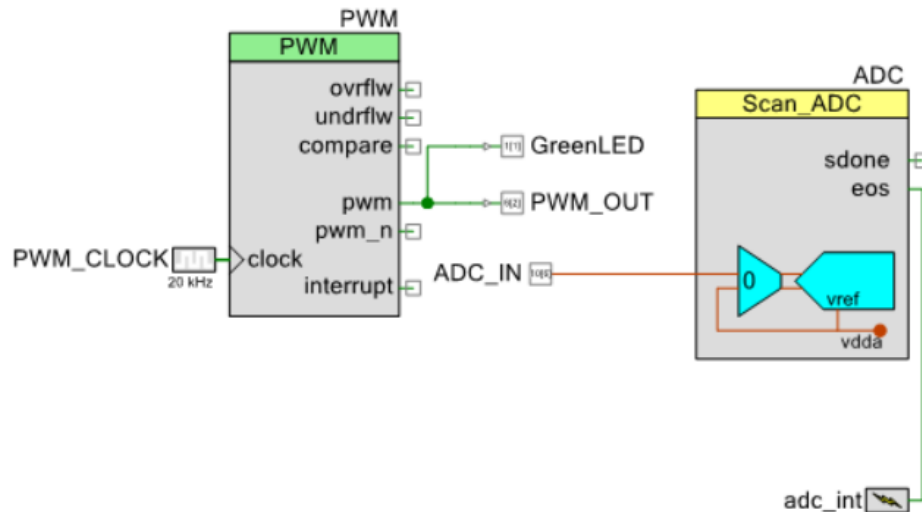


Question 2.2 (1pt) : À un duty cycle de 50%, quelle est la limite à laquelle votre œil peut voir la LED clignoter. Donner votre réponse en Hz et appuyer votre réponse d'une source fiable, i.e. un article scientifique.

Partie 3 – ADC

La dernière partie consiste à échantillonner à l'aide d'un ADC le signal généré par le PWM. Vous devez garder en mémoire une fenêtre de 2 secondes pour calculer la moyenne du signal (en mV) et le duty cycle du PWM (%).

Code 3.1 (1pt) : Ajouter une pin digitale pour la sortie du PWM (port 6 pin 2) ainsi qu'une pin analogique (port 10 pin 6), une interruption et un module ADC à votre schéma. Vous devriez obtenir ceci :



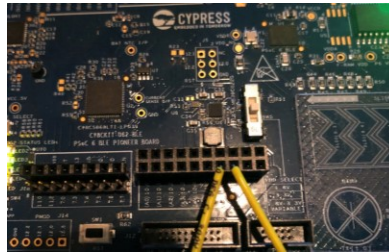
Question 3.1 (1 pts) : Quelle devrait être la fréquence d'échantillonnage de votre ADC pour mesurer le duty cycle et la moyenne du signal d'un PWM à 20Hz de 50% de duty cycle. Vous devez pouvoir mesurer le duty cycle avec une précision de 1%. Expliquez votre réponse

Configurer votre ADC comme suit :

Input Range		Vref select: <input type="text" value="Vdda/2"/> 1.650 V		12-bit code range: Volt range:			
<input type="checkbox"/> Vref bypass		Diff.: 0x000 to 0xFFFF		Vn-Vref to Vn+Vref			
Vneg for S/E: <input type="text" value="Vref"/>		<input type="text" value="S/E: 0x000 to 0xFFFF"/> <input type="text" value="0 to 2*Vref"/>					
Result Data Format		Interrupt Limits					
Diff. result format: <input type="text" value="Unsigned"/>		Compare mode: <input type="text" value="Result < Low"/>					
S/E result format: <input type="text" value="Unsigned"/>		Low (hex): <input type="text" value="200"/> High (hex): <input type="text" value="E00"/>					
Samples averaged: <input type="text" value="2"/>		Diff. value (V): 0.41 V 2.89 V					
Averaging mode: <input type="text" value="Sequential, Fixed"/>		S/E value (V): 0.41 V 2.89 V					
		Diff. avg (V): 0.41 V 2.89 V					
		S/E avg (V): 0.41 V 2.89 V					
Channels							
Number of channels: <input type="text" value="1"/>							
Ch.	En	Input mode	Avg	Minimum acq. time (ns)	Achieved acq. time (ns)	Limit interrupt	Sat. interrupt
0	<input checked="" type="checkbox"/>	Single ended	<input type="checkbox"/>	167.00	592760	<input type="checkbox"/>	<input type="checkbox"/>

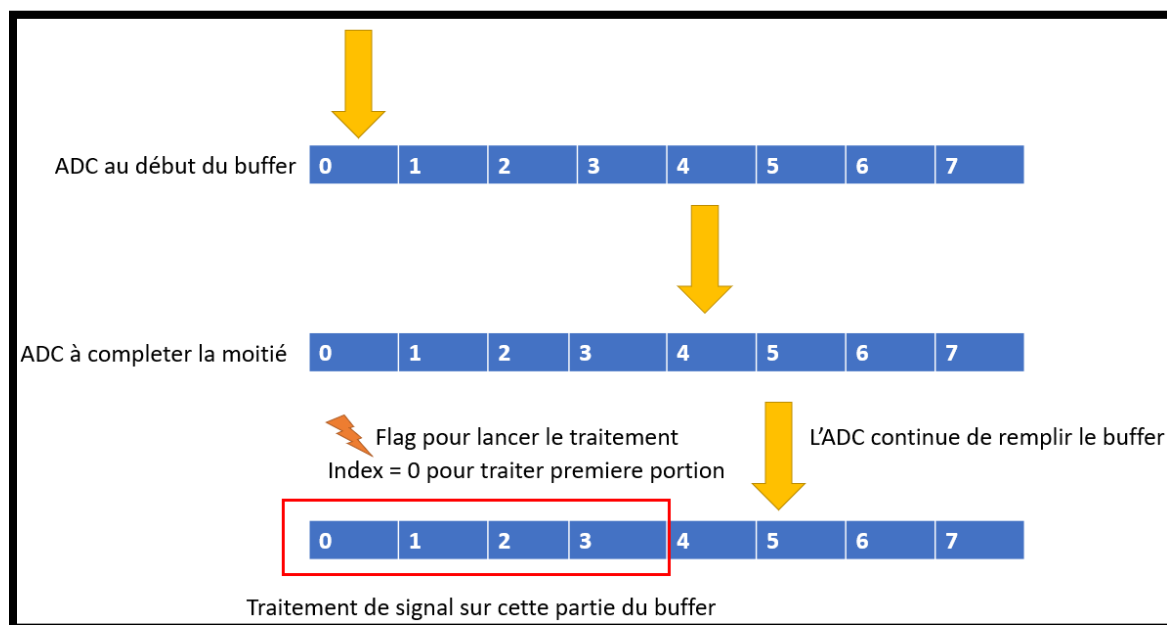
Pour référencer l'interruption `adc_int` et l'activer, vous trouverez l'information dans **les notes du cours 2**.

Vous devez connecter l'ADC à votre PWM à l'aide d'une connexion externe. Retirez votre écran et reliez la pin 6.2 et 10.6 à l'aide d'un fil de votre kit. Voir la figure suivante.



Code 3.2 (2 pts) : Écrire une fonction « `adc_handler` » qui sera appelé lorsque l'ADC a terminé sa conversion. Dans cette fonction vous devriez enregistrer la valeur de la conversion dans un buffer de type circulaire. Votre buffer circulaire devrait contenir 2s d'acquisition.

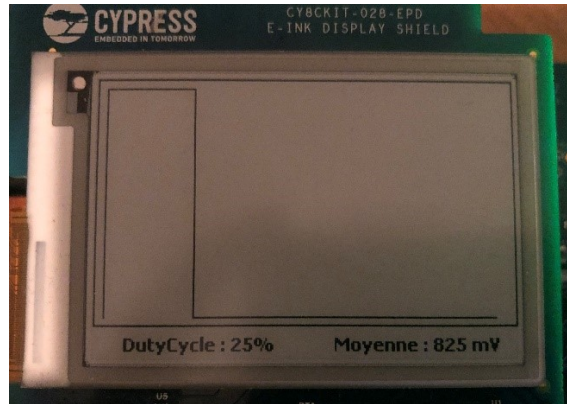
Code 3.3 (2 pts) : Ajouter un « flag » et un « index » dans votre interruption lorsque la moitié du buffer est remplie. Ce flag sera lu dans votre main et lancera un traitement de signal sur la première moitié du buffer alors que l'ADC continue de remplir le buffer. Ensuite, le flag devrait être réinitialisé et être redéclenché lorsque l'ADC retourne au premier élément du buffer. Cette fois-ci, le traitement de signal se fera sur la deuxième moitié du buffer. Pour l'instant, appeler une fonction `void traitementSignal(void)` vide. La figure suivante résume les étapes.



Question 3.2 (1 pt) : Qu'arrive-t-il si votre temps de traitement de signal est plus grand que le temps d'acquérir la moitié du buffer ?

Code 3.4 (2 pts) : Écrire une fonction qui calcule la moyenne (en mV) et le duty Cycle (en %) sur un intervalle de 1 secondes à partir du buffer circulaire. Vous pouvez approximer le duty Cycle à partir de la moyenne calculée. Appelez cette fonction lorsque le flag est déclenché.

Code 3.5 (2 pts) : Après 3 traitements de signal, afficher une période du PWM, son dutyCycle et la moyenne (en mV) sur l'écran. L'affichage devrait donc s'actualiser aux 3 secondes. Vous devriez avoir un affichage semblable à la figure suivante. **Note : si l'affichage n'est pas stable, vous pouvez arrêter votre ADC le temps de l'affichage.**



Pour tester votre programme, faites varier le duty cycle de votre PWM (25%, 50%, 75%).

Question 3.3 (1pt) : Quel est l'avantage d'utiliser un buffer circulaire ? Pourquoi n'est-il pas recommandé de traiter le buffer seulement lorsque celui-ci est plein ?

Fonctions intéressantes

Voici quelques fonctions qui pourraient vous être utiles, sans s'y limiter. Quelques fonctions sont définies dans votre projet, consultez l'entête pour comprendre leur fonctionnement. Notez que les modules ADC, CLOCK et PWM portent ces noms respectifs. **Si votre module s'appelle MONMODULEPWM**, les fonctions seront MONMODULEPWM_Start(), et ainsi de suite.

Affichage graphique :

```
void      GUI_SetBkColor    (GUI_COLOR); //pour GUI_COLOR voir GUI.h
void      GUI_SetColor      (GUI_COLOR);

U8        GUI_SetPenSize    (U8 Size);
U8        GUI_SetPenShape   (U8 Shape);
U8        GUI_SetLineStyle  (U8 Style);

void GUI_Clear              (void);
void GUI_DrawCircle         (int x0, int y0, int r);
void GUI_DrawEllipse        (int x0, int y0, int rx, int ry);
void GUI_DrawHLine          (int y0, int x0, int x1);
void GUI_DrawLine           (int x0, int y0, int x1, int y1);
void GUI_DrawPixel          (int x, int y);
void GUI_DrawPoint          (int x, int y);
void GUI_DrawRect           (int x0, int y0, int x1, int y1);
```

CLOCK & PWM:

```
__STATIC_INLINE void CLOCK_Enable(void) //Démarré un module CLOCK

void PWM_Start(void); //Démarré le module PWM
__STATIC_INLINE void PWM_DeInit(void); //DéInit le module PWM
__STATIC_INLINE void PWM_Enable(void); //Enable le module PWM
__STATIC_INLINE void PWM_Disable(void); //Arrête le module PWM
__STATIC_INLINE uint32_t PWM_GetStatus(void);
//Permet de lire et écrire les valeurs entrées dans le GUI
(paramètres du PWM)
__STATIC_INLINE void PWM_SetCompare0(uint32_t compare0);
__STATIC_INLINE uint32_t PWM_GetCompare0(void);
__STATIC_INLINE void PWM_SetCounter(uint32_t count);
__STATIC_INLINE uint32_t PWM_GetCounter(void);
__STATIC_INLINE void PWM_SetPeriod0(uint32_t period0);
__STATIC_INLINE uint32_t PWM_GetPeriod0(void);
__STATIC_INLINE void PWM_EnablePeriodSwap(bool enable);
```

ADC:

```
void ADC_Start(void); //Démarré le module ADC
void ADC_StartEx(cy_israddress userIsr); //Enregistre une int. software
__STATIC_INLINE void ADC_Stop(void); //Arrête le module ADC
void ADC_StartConvert(void); //Commence les conversions
__STATIC_INLINE void ADC_StopConvert(void); //Arrête les conversions
uint32_t ADC_IsEndConversion(cy_en_sar_return_mode_t retMode);
//Fonction bloquante qui retourne lorsque les résultats sont prêts.
__STATIC_INLINE int16_t ADC_GetResult16(uint32_t chan); //Retourne le
résultat de la conversion en fonction du canal.
```

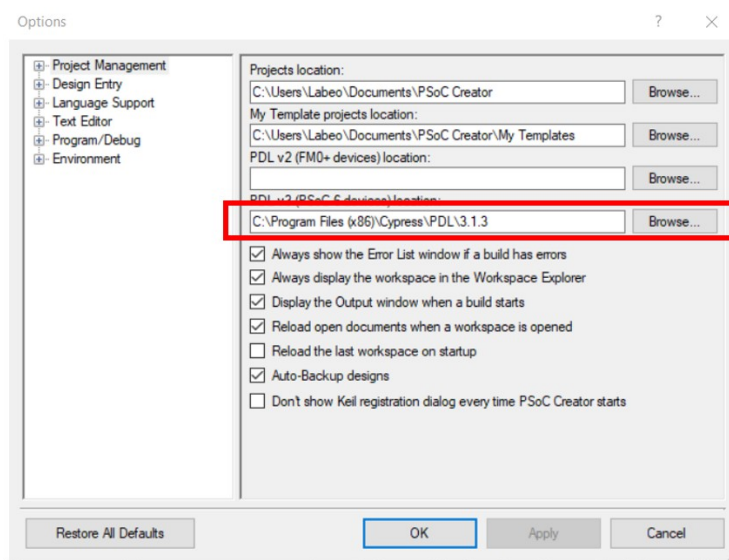

Interrupts:

```
Cy_SysInt_Init(&interruptName_cfg, Handler_Name); // Initialisation de  
l'interruption et référence vers la fonction appelée « Handler ».  
NVIC_EnableIRQ(interruptName.intrSrc); // Enable interrupt  
NVIC_ClearPendingIRQ(adc_int_cfg.intrSrc); // Clear interrupt
```

Erreur de compilation

Assurez-vous d'utiliser la PDL library 3.1.0 et plus.







Naviguez tools -> options. Vérifier que vous avez 3.1.0 et plus, sinon browser et sélectionner le bon PDL.



Rendez-vous sur le site du kit et téléchargez et exécutez

 [Download CY8CKIT-062-BLE Complete Setup](#) 

Disponible ici : <https://www.cypress.com/documentation/development-kitsboards/psoc-6-ble-pioneer-kit-cy8ckit-062-ble>

Bluetooth Low Energy Dongle).hex			
 CY8CKIT-062-BLE PSoC® 6-BLE Pioneer Kit Quick Start Guide.pdf	English	20.3 MB	12/08/2020
 CY8CKIT-062-BLE Schematic.pdf	English	2.77 MB	01/21/2020
 Download CY8CKIT-062-BLE Complete Setup 	English	605.96 MB	07/22/2018
 CY8CKIT-028-EPD Schematic.pdf	English	567.76 KB	05/03/2018
 CY8CKIT-062-BLE PSoC® 6-BLE Pioneer Kit Guide (Chinese).pdf	Chinese	2.09 MB	01/18/2021