



# Sciences et technologies de l'Industrie et du développement durable

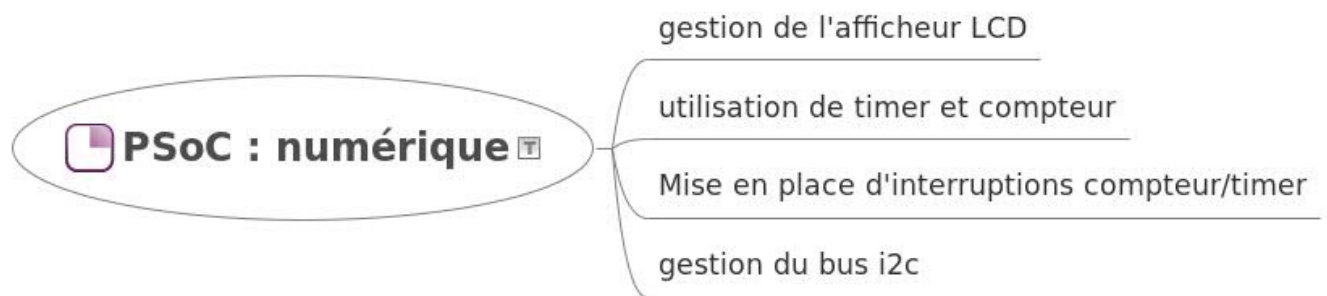
**Bac STI2D Formation des enseignants**

Denis PENARD - Jean-François LIEBAUT

**SIN 63** : Prototypage d'un traitement de l'information  
analogique et numérique (PSoc)

TP de mise en place  
d'applications  
numériques

## 1. POINTS ABORDES



## **2. MISE EN PLACE DE L'AFFICHEUR LCD POUR L'AFFICHAGE DE LA PULSATION DE LA MUSIQUE**

Lancer l'environnement de développement PSoC Designer 5.1 et créer un nouveau projet

- nom du projet : test\_lcd
- langage de programmation : langage C

Microcontrôleur (Device) : CY8C29466-24PXI (remplacer éventuellement par la référence de votre microcontrôleur).

Une fois le projet créé, nous allons rajouter le " user module " LCD que nous appellerons " afficheur ".

3 étapes seront nécessaires pour faire fonctionner l'afficheur :

- Installer le user module
- Configurer le user module
- Utiliser le user module.

### **2.1. INSTALLER LE USER MODULE**

Choisir " User Module Catalogue " du menu View ou bien sélectionner l'onglet " User Module " à droite de la fenêtre de PSoC Designer.

Choisir le module LCD de la section " Misc Digital " et le placer par un clique-droit - > place.

### **2.2. CONFIGURER L'AFFICHEUR LCD**

Sélectionner le module LCD (nom par défaut : LCD\_1) dans l'arborescence du projet (nom\_projet[Chip]/Loadable Configurations/nom\_projet - 1 User Modules.

La sélection rend accessibles les paramètres de notre afficheur, dans la zone " Parameters - LCD\_1 " sur la partie gauche de PSoC Designer.

- Renommer le module " LCD\_1 " en " afficheur ".
- Choisir le port 8 bit qui sera connecté à l'afficheur. Sur les cartes PSoC Eval1, le port utilisé est le port 2. Dans vos développements futurs, vous pouvez en choisir un autre.
- Choisir " Disable " pour l'option BarGraph. Nous n'utiliserons pas les fonctionnalités de barre graphe sur l'afficheur.

Terminer cette étape en choisissant - Generate Configuration files for "mon\_projet" Project - du menu Build ou bien en cliquant sur l'icône associée.

### 2.3. TEST DE L’AFFICHEUR MISE EN PLACE

On souhaite maintenant afficher le texte " Test LCD " sur la première ligne de l'afficheur.

- Ouvrir le fichier main.c en le sélectionnant dans l'arborescence (Workspace Explorer -> mon\_projet/Sources Files/main.c).
- Insérer les lignes suivantes dans la fonction main

```
afficheur_Start();  
afficheur_Position(0,1);  
afficheur_PrCString(" Test LCD ");
```

- Compiler et linker le projet
- Choisir la commande " Build 'mon\_projet' Project F7 " du menu Build.

### 2.4. PROGRAMMER LE PSoC

- Choisir " Program Part...CTRL+F10 " du menu Program.

Pour la programmation, connecter le programmeur miniprogrammeur à la carte d'un côté et au PC par l'USB de l'autre. Il y a deux manières de programmer :

- Mode Reset : le programmeur utilise la broche XRES du PSoC. A la fin de la programmation, le PSoC reste alimenté.
- Mode Power : le programmeur utilise une séquence sur la broche d'alimentation pour faire passer le PSoC en mode programmation. Le PSoC ne reste pas alimenté à la fin de la programmation.

Dans tous les cas, prendre son mal en patience... et réfléchir aux prochaines modifications à apporter à son programme !

### 2.5. FAQ

L'afficheur n'affiche rien :

- Le module est placé et configuré ?
- L'afficheur est-il branché ?
- Le contraste est-il correctement réglé ? (régler le petit potentiomètre à côté du LCD).

### 3. MISE EN PLACE DES INTERRUPTIONS

La mise en place des interruptions nécessite d'intervenir à plusieurs étapes du projet :

- Pendant la configuration du user module.
- Dans un fichier assembleur généré automatiquement et dans lequel la routine d'interruption pourrait être codée en assembleur. (les routines seront codées en langage C dans la quasi-totalité des projets PSoC)
- Dans le fichier main.c

#### 3.1. EXEMPLE DE MISE EN PLACE D'UNE INTERRUPTION COMPTEUR.

**Objectif** : réaliser une tâche périodiquement, lancer automatiquement une fonction lorsque le « Compteur » à fini de décompter.

**Réalisation** : incrémenter une variable toutes les secondes et l'afficher sur le LCD.

Les user modules qui permettent de « lever » une interruption, par exemple un compteur ou un timer possèdent le paramètre : « InterruptType ». Sélectionner l'événement qui provoquera l'interruption, par exemple « fin de comptage » (TerminalCount). Ce qui signifie que lorsque le compteur ou compteur arrivera à 0, une interruption sera générée.

La génération du projet entraîne la création d'un fichier en assembleur (extension **asm**) dont le nom est composé du nom du user module avec la chaîne **INT** (exemple : Counter8INT.asm).

Ce fichier possède la routine d'interruption en assembleur dont le nom ressemble à

`_Counter8_ISR:`



*ISR comme Interrupt SubRoutine.*

Enfin, ce fichier va être modifié pour accueillir la seule ligne de code assembleur de votre projet, à avoir un saut vers la routine en C. Le saut en question ne peut pas être placé n'importe où. PSoC Designer place des bannières entre lesquelles placer la ligne.

```
_Counter8_ISR:
;@PSoC_UserCode_BODY@ (Do not change this line.)
;-----
; Insert your custom assembly code below this banner
;-----
ljmp _nomdelafunctiondansleprogrammeenC
```

```
;-----  
; Insert your custom assembly code above this banner  
;-----
```



*Il y a bien un sous tiret devant l'étiquette de la fonction.*

La dernière étape concerne l'édition du fichier main.c dans lequel on va dire au compilateur que la fonction « nomdelafunctiondansleprogrammeenC » est une fonction liée à une interruption.

```
#pragma interrupt_handler nomdelafunctiondansleprogrammeenC  
void nomdelafunctiondansleprogrammeenC(void)  
{  
    //le code qui sera exécuté lorsque le compteur aura atteint 0.  
}
```

Par défaut, les interruptions sont désactivées, il faut donc :

- Les autoriser de manière générale en rajoutant la ligne suivante dans la fonction main :

```
M8C_EnableGInt ; //pas de parenthèse, il s'agit d'un #define déclaré dans  
//le fichier m8c.h
```

- Autoriser le compteur à générer l'interruption :

```
Counter8_EnableInt();
```

Dans cette partie, nous avons vu comment mettre en place une interruption pour un Compteur 8 bits.

D'autres blocs numériques/analogiques ainsi que les entrées numériques peuvent générer des interruptions. La démarche de mise en œuvre est la même.



On peut générer autant d'interruption que l'on souhaite.

## 4. MISE EN PLACE D'UNE LIAISON I2C POUR LA RECUPERATION D'UNE DONNEE SUR UNE ESCLAVE I2C.

### 4.1. MISE EN PLACE DU MODULE I2C ESCLAVE

**Objectif** : récupérer une donnée sur un esclave I2C et l'afficher sur l'écran LCD du maître.

**Ce que fait le maître** : le maître I2C lit en continu la donnée stockée dans un esclave au travers du bus I2C.

**Ce que fait l'esclave** : toutes les secondes, une variable est incrémentée d'une unité.



*Le projet de la carte afficheur « maître I2C » est fourni et commenté.*

**Réalisation** : ajout d'un module de communication I2C de type esclave sur la carte de développement.

**Tests** : Les tests requiers deux PSoC qui peuvent être installés sur le même kit de développement, l'un sur le support de programmation (le PSoC maître, car il est connecté à l'afficheur), et l'autre sur la mini plaque labdec. Les PSoC communiquent entre eux par une liaison I2C. Chaque signal (SDA et SCL) doit être connecté à 5V au travers d'une résistance de pull-up de 4,7kOhm. La référence de masse est également connectée entre les deux PSoC.

### 4.2. REALISATION DE L'ESCLAVE

La première étape met en œuvre un timer qui génère une interruption toutes les secondes. La fonction appelée chaque seconde incrémente une variable.

La deuxième étape concerne l'implémentation de l'esclave I2C.

#### 4.2.1. Mise en place de l'interruption timer

L'incrémentation de la variable toutes les secondes est réalisée dans une fonction d'interruption. L'interruption est générée par un timer.

Dans un premier temps vous allez mettre en place le timer puis l'interruption timer.



*On pourrait aussi utiliser un counter...*

Pour choisir un timer (8 bits ou 16 bits), il faut déterminer la fréquence d'horloge à appliquer, ainsi que la valeur à stocker dans le « Period Register ». Plusieurs couples - horloge/valeur du diviseur du timer - sont possibles. Le choix peut dépendre des besoins en horloge des autres blocs utilisés dans le PSoC.

Calculer la fréquence de l'horloge à appliquer au timer ainsi que la valeur à stocker dans le « Period Register » pour obtenir une impulsion sur sa sortie (TerminalCountOut), toutes les secondes.

#### 4.2.2. [Test du timer](#)

Câble en interne la sortie du timer sur une sortie du PSoC et visualiser le signal à l'oscilloscope. Vérifier que le timer est configuré pour générer une interruption toutes les secondes.

Timer8 bits <input type="checkbox"/>	Timer16 bits <input type="checkbox"/>
Clock :	_____
Capture :	low
TerminalCountOut :	A router sur une broche de sortie.
CompareOut :	none
Period :	_____
CompareValue :	0
CompareType :	less than or equal
ClockSync :	sync to sysclock
TC_PulseWidth :	Full Clock
Interrupt API :	Enable
InterruptDispatchMode :	ActiveStatus
InvertCapture :	Normal

#### 4.2.3. [Test de l'interruption](#)

Configurer le timer afin qu'il génère une interruption sur fin de comptage.

Modifier le fichier TimerXXINT.asm en ajoutant un « longjump » vers la routine d'interruption de notre programme en C.

Ecrire la routine d'interruption en C qui incrémente une variable de type char.

### 4.3. [MISE EN PLACE DU MODULE I2C ESCLAVE](#)

Choisir le module « EzI2Cs » comme Easy I2C (!) de la section " Digital Comm " et le placer par un clique-droit -> place.

#### 4.3.1. [Configurer le module esclave EzI2Cs](#)



Sélectionner le module EzI2Cs\_1 dans l'arborescence du projet.

Renommer le module « IEzI2Cs » en « I2Cs » et régler les paramètres suivants :

- Slave Address : 50
- Rom Register : Disable
- Choisir la fréquence de l'horloge I2C : I2C clock = 100K (standard).
- Choisir le port 1 (I2C port).
- Relier les signaux SDA et SCL respectivement aux broches P15 et P17 du PSoC.

Terminer cette étape en choisissant - Generate Configuration files for 'mon\_projet' Project - du menu Build.

#### 4.3.2. [Utiliser le module I2C](#)

Nous allons maintenant ajouter le code en langage C permettant d'initialiser le module I2Cs puis de répondre aux demandes du maître I2C.

- Ouvrir le fichier main.c et insérer les lignes suivantes :

```
unsigned char mon_compteur;

void main(void)
{
    // initialisation du module I2Cs.
    // La fonction I2C_SetRamBuffer configure l'adresse en RAM de la structure
    // de données et définit sa taille. Elle est à appeler avant la fonction
    // I2Cs_Start()
    I2Cs_SetRamBuffer(sizeof(mon_compteur), sizeof(mon_compteur), (BYTE *)(&
mon_compteur));
    I2Cs_Start();

    mon_compteur = 0x12;
    //Terminer le programme avec une boucle sans fin :
    while(1);
}
```

- Compiler et linker le projet avec la touche raccourcis F7 ou la commande " Build 'mon\_projet' Project " du menu Build.
- Programmer le PSoC.

#### 4.3.3. [Tester la communication](#)

Branchement des cartes :

- Relier la broche P10 et P11 du maître respectivement sur P15 et P17 de l'esclave.

- Connecter les signaux SDA et SCL à 5V au travers de deux résistances de pull-up de 4,7kOhm (la valeur de la résistance peut varier de 1,5kOhm à 6kOhm).
- Relier la référence de masse des cartes PSoC.



*Les broches P10 et P11 servent à la programmation du PSoC sur la carte d'évaluation. Si vous avez fait le montage complet, la programmation du PSoC maître ne se fera pas si la broche P10 est connectée au PSoC esclave.*

Rajouter la fonction d'interruption qui incrémente la variable mon\_compteur.

#### 4.3.4. Test de la communication I2C avec une variable de type INTEGER (int).

Que faut-il modifier dans le projet du maître I2C pour récupérer une variable de type integer sachant que la communication I2C ne sait manipuler que des octets ?



*Plusieurs solutions sont possibles :*

La première consiste à « découper » l'INTEGER en deux octets qu'on stocke dans deux variables de type CHAR (l'octet MSB et l'octet LSB). Le maître va donc lire deux octets à la suite et les réassembler.

Exemple :

```
#include <stdio.h>
#include <stdlib.h>

//On déclare un integer :
unsigned int uiDistance ; //le ui devant distance rappel le type.
//et deux octets
unsigned char ucDistanceMSB ; //l'octet de poid fort
unsigned char ucDistanceLSB ; // l'octet de poid faible.

int main()
{
    //On affecte une valeur à uiDistance : 0x1234 ;

    uiDistance = 0x1234 ;
    //On charge ucDistanceMSB et ucDistanceLSB avec repectivement 0x12 et
    0x34
    ucDistanceMSB = uiDistance>>8;
    ucDistanceLSB = uiDistance;
    printf("uiDistance : %X\n", uiDistance);
    printf("uiDistanceMSB : %X\n", ucDistanceMSB);
    printf("uiDistanceLSB : %X\n", ucDistanceLSB);

    //Assemblage
    uiDistance = (uiDistanceMSB<<8)+ ucDistanceLSB;
    printf("uiDistance : %X\n", uiDistance);
```

```
    return 0;
}
```

L'autre technique plus élégante consiste à utiliser une union entre la distance (codée sur 2 octets) et un tableau de 2 octets.

En C, les unions permettent la superposition en mémoire de variables de types différents. Une union permet ainsi d'interpréter de différentes manières un même emplacement en mémoire.

L'utilisation des variables déclarées dans l'union ressemble à l'utilisation d'une structure en C.

Voici le même programme que précédemment, mais qui utilise une union.

```
#include <stdio.h>
#include <stdlib.h>

//on déclare une union que j'appelle union_distance et qui possède 2
//variables :
union {
    unsigned int uiDistance;
    unsigned char ucdist_tab[2]
} union_distance ;
//on peut aussi écrire dist_tab[sizeof(distance)] à la place de
//ucdist_tab[2]

int main()
{
    //On affecte une valeur à uiDistance : 0x1234 ;
    union_distance.uiDistance = 0x1234 ;
    printf("uiDistance : %X\n", union_distance.uiDistance);
    printf("ucDistance LSB : %X\n", union_distance.ucdist_tab[0]); //on
affiche 0x34, il faut stocker les LSB dans union_distance.ucdist_tab[0]
    printf("ucDistance MSB : %X\n", union_distance.ucdist_tab[1]); //on
affiche 0x12, il faut stocker les MSB dans union_distance.ucdist_tab[1]

    return 0;
}
```