INFO2 : INFORMATIQUE EMBARQUEE
TP4 : Mise en œuvre d'un afficheur LCD

1 Objectifs

Il s'agit de mettre en œuvre l'afficheur LCD présent sur la carte d'évaluation EasyPIC v7. Celui-ci est à 2 lignes de 16 caractères (chaque caractère est une matrice de 5 pixels de large sur 7 pixels de haut) et il est géré par un classique contrôleur Hitachi HD44780 à l'arrière. Pour commander l'afficheur, et donc lui faire afficher des informations, le PIC de la carte d'évaluation communique avec le contrôleur HD44780.

2 Introduction

DOC_LCD_HD44780.pdf, une documentation du contrôleur HD44780, est présente dans le répertoire **Ressources INFO2\Documentation\LCD**. Cette documentation est relative à un afficheur de 4 lignes de 20 caractères mais les principes sont les mêmes que ceux de l'afficheur avec lequel on travaille.

- Ouvrez cette documentation.

easypic_v7_manual_v104.pdf, la documentation de la carte d'évaluation, est présente dans le répertoire **Ressources INFO2\Documentation\Carte EasyPic v7**.

- Ouvrez cette documentation à la page 24.
- Confrontez les deux documentations pour répondre aux questions suivantes :
    - Le PIC communique-t-il avec le HD44780 en mode 4 bits ou en mode 8 bits ?
    - Quel est le niveau permanent de la broche $R/\overline{W}$? Le PIC pourra-t-il alors lire des informations en provenance du contrôleur HD44780 ?

La réponse à la question précédente a une conséquence importante. Normalement, avant d'envoyer la moindre commande au contrôleur HD44780, le PIC doit s'assurer que celui-ci n'est pas occupé, donc faire une lecture de son *busy flag*. Vous vous êtes rendus compte que, vus les choix des concepteurs de la carte EasyPIC v7, aucune lecture n'est possible. Après chaque commande envoyée, il faut donc temporiser suffisamment longtemps afin que le contrôleur HD44780 traite correctement la commande reçue. Le souci vient du fait que les temporisations à générer dépendent de la commande à envoyer (effacement de l'écran, envoi d'un caractère, etc.)…

Remarque fondamentale : Même si nous allons utiliser une bibliothèque « haut-niveau » de gestion de l'afficheur masquant ces aspects pénibles de programmation, il ne faut pas perdre de vue qu'il y a beaucoup de temps passé à attendre. Les choses ne sont pas instantanées. Il ne faut pas hésiter à faire une simulation temporelle si nécessaire.

3 Projet TP4A : « Hello world ! » de l'afficheur

3.1 Cahier des charges et objectifs

A l'issue du programme à écrire, on doit voir sur l'afficheur le message ci-dessous :

		Position x															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Position y	0	**H**	**e**	**l**	**l**	**o**											
	1							**w**	**o**	**r**	**l**	**d**	**!**				

Ce premier projet permet de valider que l'afficheur fonctionne, de vérifier que l'on est en mesure de lui envoyer des caractères un par un et de les placer où l'on souhaite sur l'écran.

3.2 Mise en place de la bibliothèque de gestion de l'afficheur

Le compilateur C18 vient accompagné de toute une série de bibliothèques. Notamment, il en existe une ayant pour fichier d'en-tête **xlcd.h** permettant de gérer un contrôleur HD44780 connecté au PIC. Cette bibliothèque a été légèrement modifiée pour être adaptée à la carte EasyPIC v7, notamment au fait qu'aucune lecture du contrôleur HD44780 ne soit possible.

- Créez un projet de nom **TP4A**. Ajoutez-y un fichier **TP4A.c** commençant comme ceci :

```
/* Auteur       : OL
 * Creation     : 31/03/14
 * Derniere MAJ : 01/04/14
 * IDE          : MPLAB X
 * Compilateur  : C18
 * Programme pour la carte EasyPic v7 avec un PIC18F4520 (DIP40)
 * Test basique de l'afficheur LCD
 */

#include <p18f4520.h>
#include "../configuration_bits.h"
#include <delays.h>
#include "INFO2_xlcd.h"

void main (void) {
      // initialisation de l'afficheur et affichage du message :



      while(1);
}
```

- COPIEZ dans votre répertoire de projet **`TP4A.X\`** les huit fichiers de la bibliothèque (un fichier H et sept fichiers C) présents dans **`Ressources INFO2\Fichiers\LCD\`**. Pour information, tous les noms de fichiers de la bibliothèque LCD modifiée commencent par le préfixe **`INFO2_`**.

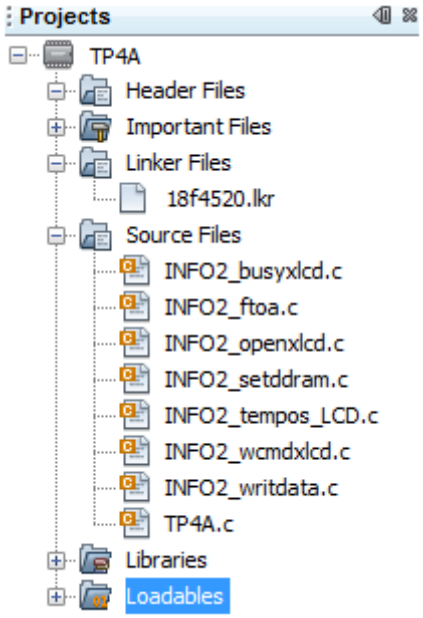
Remarque : Le fichier **`INFO2_xlcd.h`**, comme tout fichier H (*header*) qui se respecte, comprend des **constantes** et des **prototypes de fonctions**. Ces fonctions sont celles écrites dans les sept fichiers C copiés. L'ensemble des constantes et des fonctions permettent de gérer le LCD.

Remarque : Ces huit fichiers ne doivent pas être modifiés !

Dans les étapes ci-dessous, il faut cibler les fichiers copiés dans votre répertoire de projet, pas ceux du répertoire **`Ressources INFO2\Fichiers\LCD\`** :

- Déclarez les sept fichiers d'extension **`c`** comme faisant partie des *Source Files* du projet : clic droit sur Source Files puis Add Existing Item….
- Enfin, comme d'habitude, n'oubliez pas de préciser **`18f4520.lkr`** comme faisant partie des *Linker Files*.

A la fin, vous devez avoir quelque chose comme ceci :



3.3 Fin de l'écriture de TP4A.c

Pour cette partie, vous aurez besoin de consulter la partie 3.2 de **`MPLAB_C18_Libraries_51297f.pdf`** (documentation des bibliothèques du compilateur C18).

Les fonctions qui vous seront utiles sont : **`OpenXLCD()`**, **`SetDDRamAddr()`** et **`WriteDataXLCD()`**.

Remarque : **`SetDDRamAddr()`** permet de choisir où va être déposé le prochain caractère que l'on va envoyer. Exploitez **`DOC_LCD_HD44780.pdf`** pour comprendre la valeur du paramètre à passer à cette fonction.

Remarque : Quand on a envoyé un caractère et que le suivant doit se trouver immédiatement à sa droite, il n'est pas besoin d'utiliser **`SetDDRamAddr()`** car le curseur se déplace automatiquement d'un cran à droite après l'envoi d'un caractère.

- Finissez d'écrire votre programme afin de respecter le cahier des charges.

4 Projet TP4B : Utilisation de fonctions évoluées d'affichage

4.1 Cahier des charges et objectifs

A l'issue du programme à écrire, on doit voir sur l'afficheur le message ci-dessous. Pour cela, on utilisera la fonction d'affichage de chaîne **`fprintf()`**. Sur la deuxième ligne le nombre évolue entre -150 et +150 ; il s'incrémente toutes les 100 ms. Quand la valeur maximale est atteinte, on repasse à la valeur minimale.

		Position x															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Position y	0	**W**	**h**	**a**	**t**	**'**	**s**		**u**	**p**	**,**		**d**	**o**	**c**	**?**	
	1	**i**	**=**	**-**	**1**	**3**	**6**										

On l'a vu, pour l'instant nous parvenons à envoyer des caractères un par un. Cela peut vite s'avérer fastidieux. Dans ce projet, nous allons apprendre à utiliser **`fprintf()`**, une variante de **`printf()`**, dans le but d'envoyer facilement des chaînes de caractères.

4.2 Mise en place du projet

- Pour ce projet et les suivants, vous n'allez pas recréer un projet car vus les fichiers à déclarer, c'est un peu long. Vous allez retirer **`TP4A.c`** de votre projet par un clic-droit sur celui-ci, suivi d'un Remove From Project. Ensuite vous créerez un fichier **`TP4B.c`** que vous ajouterez au projet comme d'habitude.

4.3 Présentation de fprintf()

En C « classique », quand on veut réaliser un affichage, on utilise la puissante fonction **`printf()`**[1]. Par exemple :

```
// Affichage d'une chaîne de caractères constante :
printf("Comment ça va ?");

// Affichage de l'entier signé n sur au moins 6 caractères (complété avec des espaces à gauche si besoin):
printf("%6d", n);

// Affichage de la chaîne (tableau) de caractères de nom chaineDeTest :
printf("%s", chaineDeTest);
```

La fonction **`printf()`** est une fonction de sortie vers le flux (*file* ou *stream*) de sortie standard appelé **`stdout`**. En général, il s'agit d'un écran VGA. Bien-sûr, il n'y a pas d'écran VGA relié au PIC. **`stdout`** correspond à la liaison RS232 du PIC auquel il conviendrait de connecter un émulateur de terminal pour visualiser des chaînes de caractères. Ce sera pour plus tard, nous souhaitons utiliser pour l'instant l'écran LCD comme dispositif de visualisation de sortie.

Comment faire pour envoyer les données vers l'écran LCD avec le confort de **`printf()`** ? Il faut utiliser **`fprintf()`**. Le premier **`f`** signifie *file*. Nous allons écrire dans un *file* (ou *stream* ou flux) défini par nous-mêmes.

- Ouvrez la documentation des bibliothèques à la partie 4.7 CHARACTER OUTPUT FUNCTIONS.
- Trouvez le fichier d'en-tête à inclure en haut de votre fichier **`TP4B.c`** afin de pouvoir utiliser **`fprintf()`**.
- Relevez le prototype de la fonction élémentaire d'envoi d'UN caractère vers le flux **`_H_USER`** défini par l'utilisateur.
- Implémentez au début de **`TP4B.c`** cette fonction, après les directives préprocesseur (tout ce qui commence par **`#`**). Celle-ci doit envoyer vers l'écran LCD l'unique caractère passé en paramètre.
- Ensuite **`fprintf()`** est utilisable. Celle-ci « se débrouille » (ce n'est pas à nous de l'écrire) pour afficher correctement sur l'écran LCD les messages qu'on lui passe grâce à des appels successifs à la fonction élémentaire d'envoi d'UN caractère. Par exemple :

[1] Le **`f`** de **`printf()`** signifie *formatted*. **`printf()`** sert à effectuer des sorties formatées.

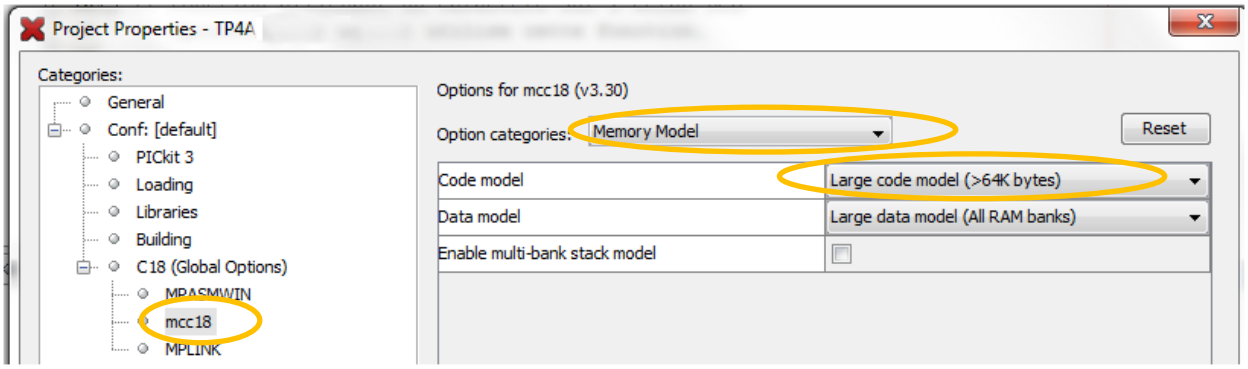
```
// Affichage d'une chaîne de caractères constante :
fprintf(_H_USER, "Comment ça va ?");

// Affichage de l'entier signé n sur au moins 6 caractères (complété avec des espaces à gauche si besoin):
fprintf(_H_USER, "%6d", n);

// Affichage de la chaîne (tableau) de caractères de nom chaineDeTest :
fprintf(_H_USER, "%s", chaineDeTest);
```

4.4 Réglage du « modèle de mémoire »

Certaines bibliothèques de fonctions nécessitent un « modèle de mémoire large ». Comme en témoigne **`MPLAB-C18-Getting-Started_51295f.pdf`** :

"When using the standard libraries by including **`stdio.h`**, **`usart.h`**, *etc., the large code model should be selected for the project. Otherwise a 2066 warning is generated when using* **`printf`**, **`puts`**, *etc."*

- Pour effectuer ce réglage : File / Project Properties puis :



4.5 Validation du travail

- Ecrivez votre programme et faites valider votre travail par l'enseignant.

5 Projet TP4C : Gestion du type float, utilisation de la bibliothèque des fonctions mathématiques

5.1 Cahier des charges et objectifs

On souhaite afficher le résultat du calcul d'un sinus comme dans la photographie ci-dessous :

0.7854 (sur la photographie) correspond à $\frac{\pi}{4}$. Il faut donc écrire en début de programme :

```
#define PI 3.1415926
```

Puis il faut déclarer dans **`main()`** un réel et l'affecter :

```
float angle = PI / 4;
```

Cet exercice comporte deux difficultés : convertir des réels en chaînes de caractères et effectuer un calcul de sinus. On donne quelques pistes ci-après.

5.2 Conversion d'un réel en chaîne de caractères

Dans un monde idéal, il suffirait d'écrire quelque chose comme :

```
fprintf(_H_USER, "sin(%f)=%f", angle, sin(angle));
```

Malheureusement, les concepteurs des bibliothèques du compilateur C18 n'ont pas implémenté l'affichage au format **`%f`** (sans doute trop compliqué pour un si petit processeur).

Heureusement, une fonction **ftoa()** (*Float TO Alphanumeric*) a été écrite par des enseignants. Elle est implémentée dans **INFO2_ftoa.c** (que nous avons déjà mis dans les Source Files à la **partie 3.2**) mais nous ne l'avons pas encore utilisée.

Intéressons-nous aux arguments de la fonction **ftoa()** :

ftoa (float x, char str[], char prec, char format)

- **x** est le réel à convertir en chaîne de caractères,
- **str** est le nom du tableau (= chaine) de caractères qui accueillera le réel converti,
- **prec** est la précision (le nombre de chiffre après la virgule),
- **format** est un paramètre valant **'f'** pour un affichage standard ou **'s'** pour un affichage scientifique.

Par exemple si on a :

```
char myString[15];
float f = 5000 / 3.0;
ftoa(f, myString, 4, 'f');
```

myString correspondra à la chaîne « 1666.6666 ».

Alors que si on a :

```
char myString[15];
float f = 5000 / 3.0;
ftoa(f, myString, 4, 's');
```

myString correspondra à la chaîne « 1.6666E3 ».

- On vous conseille de faire ces essais d'affichage de réel sur l'afficheur LCD avant de poursuivre ! Une fois la chaîne créée, il faut bien-sûr l'afficher avec **fprintf()**.

5.3 Utilisation d'une fonction mathématique

- Consultez le chapitre Math Libraries du document **MPLAB_C18_Libraries_51297f.pdf** afin de voir comment calculer un sinus. Quel **#include <xxx.h>** faut-il ajouter au début de **TP4C.c** afin d'utiliser la fonction **sin()** ?

5.4 Validation du travail

- Ecrivez votre programme et faites valider votre travail par l'enseignant. Est-ce que cela fonctionne encore quand vous initialisez la variable **angle** à **PI / 3** ?

5.5 Evaluation du temps de calcul d'un sinus

Combien de temps prend le calcul d'un sinus sur un microcontrôleur dépourvu d'une *Floating Point Unit* ?

Pour répondre à cette question, vous allez utiliser le simulateur déjà rencontré lors d'une précédente séance.

- Dans Project Properties, choisissez le simulateur plutôt que le PICkit3. N'oubliez pas que vous avez un quartz à 8 MHz.
- Mettez en commentaire tout ce qui concerne la gestion matérielle (LCD, etc.) et encadrez par deux *breakpoints* l'instruction de calcul du sinus. Préparez la *stopwatch*. Notamment, remettez-la à zéro avant d'exécuter le calcul du sinus.
- Au moment d'exécuter le calcul du sinus, choisir Step Over plutôt que Step Into. Cela a pour effet d'exécuter toute la fonction **sin()** plutôt que de rentrer pas à pas dedans.
- Combien de temps prend ce calcul ? Et en nombre d'*instruction cycle*, cela fait combien ? Est-ce beaucoup ?

6 Projet TP4D : Affichage d'une tension

6.1 Cahier des charges et objectifs

L'affichage de la tension disponible sur le potentiomètre P1 connecté à RA3 doit se faire en continu sur l'afficheur LCD de la façon suivante :

- Sur la première ligne de l'afficheur LCD, on voit le résultat brut de la conversion, c'est-à-dire un entier entre 0 et 1023.
- Sur la deuxième ligne de l'afficheur LCD, on voit la tension en Volts suivi de « V », son unité.

7 Projet TP4E : Affichage d'une température

7.1 Cahier des charges et objectifs

L'affichage de la température issue du capteur LM35connecté à RE2 doit se faire en continu sur l'afficheur LCD de la façon suivante :

- Sur la première ligne de l'afficheur LCD, on voit le résultat brut de la conversion, c'est-à-dire un entier entre 0 et 1023.
- Sur la deuxième ligne de l'afficheur LCD, on voit la température en °C suivi de « °C », son unité.