Labo 3 – L'interface au matériel – afficheur à 7 segments/LCD
CEG3536 - Architecture des Ordinateurs II
University of Ottawa

Professor : **Mohamed Ali Ibrahim**
AE : Hamidou Nouhoum Cisse
Céline Girard

Rapport de laboratoire écrit par :
Brian Makos|300194563
Scott Makos|300194574

Date du Laboratoire :

Novembre 18 2022

Introduction :

Objectifs :

Ce laboratoire a comme but de nous introduire à l'interface du matériel au Motorola 9S12DG256. Afin de faire ceci, nous allons réaliser une application d'un dispositif d'affichage à 7 segments et l'interface à un afficheur à cristaux liquides (LCD). De plus, nous aurons besoin de modifier deux fichier assembleur (delay.asm et keypad.asm) afin qu'ils soient des fichier header. Par la suite, nous aurons besoin de développer deux logiciel C (SegDisp.c et LCD_Display.c). Ces quatre fichiers seront ensuite ouvert dans Code Warrior afin de pouvoir les exécuter et utiliser la carte Debug-12.

Matériel et composantes utilisés :

- CodeWarriors
- Carte Debug-12

Conception materiel/logiciel :

1. Présentez un circuit qui montre comment le matériel est branché aux ports du microcontrôleur. Montrez le clavier et les afficheurs

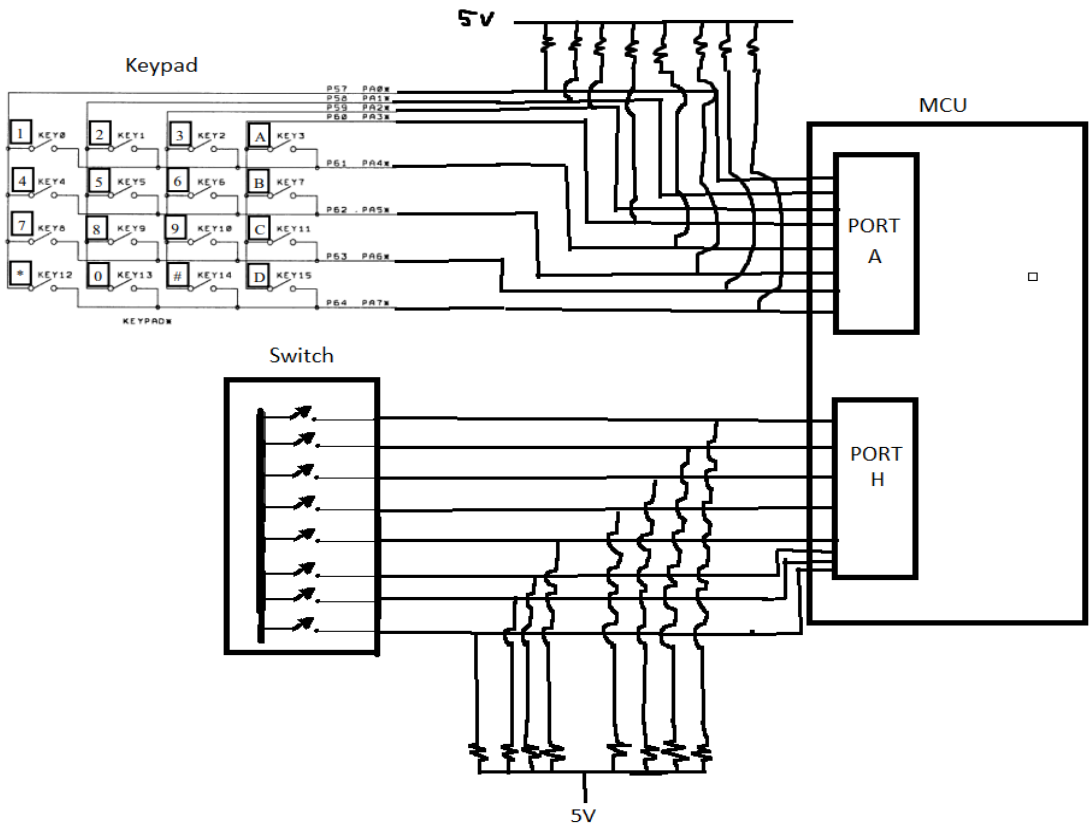

Figure 1 : Circuit qui demontre comment le matériel est branché au port du microcontrôleur (MCU)

Le clavier est connecté aux port A. Le clavier fonctionne comme une matrice, où, chaque colonne et chaque rangée possèdent une valeur d'entrée. Ces valeurs sont ensuite vérifiées grâce à une basse tension donnée par les 8 fils avec résistance qui sont connectés aux 5V. En autre mots, si la touche 1 est activer, la colonne 1 et la rangée 1 sont activé. Par la suite, ces deux fils passent par une basse tension afin de vérifier qu'ils ont été appuyés. Une fois vérifier, ces deux valeurs de 1 sont envoyées aux port A.

Par la suite, le « Switch » est connecté aux port H du MCU, celui-ci a comme but d'observer l'états des commutateurs des portes et fenêtres surveillés par le système d'alarme.

2. La conception de logiciel (selon les lignes directrices données) des modules LCD Display et Segment Display. Soyez le plus complet possible. Il n'est pas nécessaire d'inclure du pseudo-code, mais décrivez toutes les fonctions incluses dans vos modules. Divisez les fonctions en deux groupes : les « Points d'entrée » qui sont les fonctions C appelées par d'autres modules et les « Fonctions locales » qui sont les fonctions internes du module. Il N'est PAS nécessaire d'inclure la conception des modules KeyPad et Delay.

Code Partie SegDisp :

```
void initDisp(void){

    clearDisp();
    DDRB = 0xFF; //init PORTB to output
    DDRP = 0xFF; // init PORTP to output
}
```

Void initDisp() :

```
/*-------------------------------------------
Fonction: initDisp

Initialise le matériel (port B et port P) branché aux afficheurs à 7-segments.
Il initialise aussi les afficheurs en blanc (aucun segment allumé)
-------------------------------------------*/
```

Cette fonction commence en faisant appel à la fonction clearDisp() pour s'assurer que l'afficheur vide. Il nous permet aussi d'initialiser les ports B et P. On met leurs valeurs a 0xFF.

```
void setCharDisplay(char cha, byte numdisp){

    byte check = 0x00;

    if (cha=='0'){
        check = 0x3F;
    }
    else if (cha == '1'){
        check = 0x06;
    }
    else if (cha == '2'){
        check = 0x5B;
    }
    else if (cha == '3'){
        check = 0x4F;
    }
    else if (cha == '4'){
        check = 0x66;
    }
    else if (cha == '5'){
        check = 0x6D;
    }
    else if (cha == '6'){
        check = 0x7D;
    }
    else if (cha == '7'){
        check = 0x07;
    }
    else if (cha == '8'){
        check = 0x7F;
    }
    else if (cha == '9'){
        check = 0x6F;
    }
    else if (cha == 'a'){
        check = 0x77;
    }
    else if (cha == 'b'){
        check = 0x7D;
    }
    else if (cha == 'c'){
        check = 0x39;
    }
    else if (cha == 'd'){
        check = 0x5E;
    }
    else if (cha == '#'){
        check = 0x70;
    }
    else if (cha == '*'){
        check = 0x46;
    }

    digit[numDisp] = check;
```

void setCharDisplay(char, byte):

```
/*-----------------------------------------
Fonction: setCharDisplay

Une fonction qui ajoute un caractère (identifier avec son
code ASCII) à afficher. Réservez 4 octets en mémoire (un tableau) pour contenir soit des caractères ou
codes pour afficher les caractères sur les afficheurs correspondants. Lorsque la fonction est appelée,
deux arguments sont fournies, le caractère à afficher (premier argument) et un numéro d'afficheur
(débute à 0) pour indiquer sur lequel des afficheurs le caractère doit apparaitre.
-----------------------------------------*/
```

Cette fonction prend deux arguments un charactère et une byte. Le charactère s'agit du charactère qui serait afficher et le deuxième s'agit d'une byte qui nous indique sur quel afficheur le charactère devrait

apparaitre. Cette fonction traduit le charactère désiré et le transforme en code ASCII. Ces charactères sont ensuite mis dans un array à la position indiquer par la byte. On filtre les code ASCII avec des ifs et else ifs

```
void segDisp(void){
    int a,b;

    for (a = 0; a < 5; a++)
    {
        for(b  = 0; b < 4; b++)
        {
            PTP = PTPDir[b];
            PORTB = digit[b];
            delayms(5);
        }
    }
}
```

```
/*-------------------------------------------
Fonction: segDisp
Une fonction qui met à jour les afficheurs pour une période de temps en
100 ms. Ceci permet de la fonction appelante de regagner le contrôle périodiquement pour lui
permettre de compléter d'autres tâches tel que la vérification du clavier
-------------------------------------------*/
```

Cette fonction contient deux boucles. La première boucle permet de créer un délai de 5ms pour s'assurer de bien afficher le charactère sur l'afficheur et pour permettre une vérification facile pour l'utilisateur. La deuxième boucle permet d'assigner au PTP la valeur désirée, comme ceci doit être fait en étape d'un, la boucle se répète 4 fois. Cette boucle met aussi les valeurs dans PORTB qu'on désire afficher.

void clearDisp(void):

```
void clearDisp(void){
    PTP = 0x0e;
    PORTD = 0;
    PTP = 0x0d;
    PORTD = 0;
    PTP = 0x0b;
    PORTD = 0;
    PTP = 0x0e;
    PORTD = 0;

}
```

```
/*-----------------------------------------
Fonction: clearDisp
 Cette fonction met les afficheurs en blanc
-----------------------------------------*/
```

En commençant avec le premier afficheur, on assigne au PORTB une valeur qui va être affiché à 0 ainsi, les afficheurs seront vides. On répète pour chaque afficheur.

Code Partie lcdDisp

```
void initLCD(void){
    lcd_init();
}
```

void initLCD(void):

```
/*
Fonction : initLCD      Parametre : Aucun       Return : Aucun

Cette fonction initialize le materiel LCD avec une fonction present fans LCD.ASM .
*/
```

La description en vert explique le code.

```
void printLCDString(char *str,byte linenumb){

    if(linenumb ==0){
        set_lcd_adresse(0x00); //address de ligne 0
        type_lcd(str); // imprime le string voulu sur le LCD
    }

    if(linenumb ==1){
        set_lcd_adresse(0x40); //address de ligne 1
        type_lcd(str); // imprime le string voulu sur le LCD
    }
}
```

void printLCDStr(char *, byte):

```
/*
Fonction : printLCDString
Parametres :
          1. str
           Ce parametre est le "string" qui sera afficer sur le LCD
          2. linenumb
           Ce parametre est soit 0 (premier ligne du afficheur LCD) ou 1
           (deuxieme ligne du afficheur LCD). Ill sera utiliser pour choisir
           la ligne sur laquelle imprimer le 'string'
Return  : Aucun

*/
```

Cette fonction affiche une chaine sur une ou deux lignes de l'afficheur LCD. L'adresse de la chaîne à être afficher est passée dans le premier argument tandis que le deuxième argument est soit 0, soit 1 pour identifier la première ou la deuxième ligne respectivement.

On commence en vérifiant quelle ligne utiliser (à l'aide de if). Ensuite si la ligne est 0 on assigne une valeur de 0x00 et si la ligne a une valeur de 1 on assigne une valeur de 0x40. Ceci affichera plus tard une chaine de charactère.

Conclusion :

En conclusion, le laboratoire a été moyennement un succès. En raison de notre absence médicale, nous n'avons pas pu bien se préparer pour la démonstration. Cependant, nous avons pu terminer notre code et notre rapport. Lors du laboratoire, nous avons réalisé une application d'un dispositif d'affichage à 7 segments et l'interface à un afficheur LCD. De plus, nous nous sommes familiarisés avec le matériel au Motorola 9S12DG256.

Difficultés/Défis :

En raison de notre absence médicale lors des deux semaines de préparations, nous avions eu beaucoup de problèmes lorsqu'il vient aux démonstrations de nos fichiers. Par conséquence, le TA à mentionner qu'il évaluera notre rapport et non notre démonstration.