5 Commande d'afficheur alphanumérique à Cristaux liquides a partir d'un clavier matriciel

Description générale

- **Date du TP :** 9ième semaine ;
- **Durée du TP :** 3h (2 semaines) ;
- **Matériels nécessaires** : Carte d'experimentation, microcontrôleur PIC 16f877, programmateur, des composants électroniques ;
- **Logiciels utilisés :** Complilateur C PCW, Simulateur ISIS, IC Writer ;
- **Public cible :** 4ième niveau d'études de technicien supérieur en informatique industrielle ;
- **Partie du cours en rapport avec le TP :** Classification architecturale des systèmes temps reel, les microcontrôleurs.

Objectifs

- Manipuler les claviers et les écrans LCD ;
- Approfondir les acquis en programmation C des microcontrôleurs PIC ;
- Maitrise des architectures des microcontrôleurs Microchip PIC16.

Pré requis

Notions générales sur :
- L'électronique ;
- Algorithmique ;
- Programmation en C ;
- Les microcontrôleurs.

5.1 Introduction

Lorsque l'on décide de programmer une application a base de microcontrôleur PIC en langage C, la partie la plus difficile de cette programmation consiste à gérer tout ce qui a trait aux entrées/sorties et, plus géneralement, aux ressources internes du microcontrôleur. En effet, si le langage C des PIC ne diffère pas du C " standard " ou ANSI C ce qui permet d'utiliser tous les algorithmes classiques et largement diffusés, la gestion des périphériques internes et des interruptions est propre au monde des microcontrôleurs. La manipulation suivant va donc vous proposer un certain nombre d'exemples de programmes de gestion de ces ressources internes ; exemples que nous avons choisis de façon à balayer le plus largement possible ces dernières.

5.2 Entrées parallèles en matrice

De nombreuses applications ont besoin de disposer d'un clavier d'entrée d'informations, même si ce dernier n'a que peu de touches. Ainsi, un composeur téléphonique aura besoin de douze touches pour simuler un cadran de téléphone, un programmateur domestique aura besoin de touches d'entrée de l'heure et de la date, et ainsi de suite. Lorsque le nombre de touches nécessaires est tres faible, celles ci peuvent être réalisées sous forme de poussoirs câblés et le problème est résolu.
Lorsque le nombre de touches devient plus important, il faut adopter une autre solution. En effet, avec la méthode une touche une patte d'entrée, on sature vite le mieux pourvu des microcontrôleurs dès lors que l'on doit scruter un clavier un peu conséquent. On adopte alors géneralement la solution du clavier câblé en matrice. La figure 5.1 présente son principe de câblage dans le cas d'un clavier à douze touches mais il est extensible à l'infini.

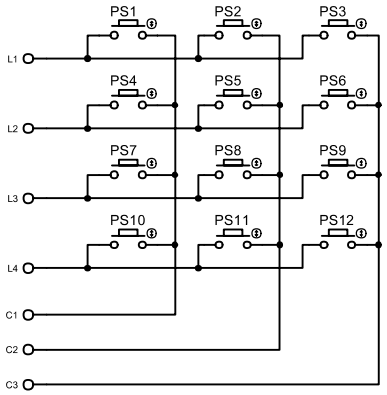

FIG. 5.1 – Principe de câblage en matrice d'un clavier

Les touches du clavier sont placées à l'intersection des lignes et des colonnes du quadrillage

réalisé par les deux groupes de fils. L'appui sur une touche court-circuite donc une ligne avec une colonne. C'est cette situation qui est mise à profit pour localiser la position de la touche grâce a un logiciel de contrôle fort simple qui fonctionne de la façon suivante. À l'instant t, les lignes sont placées en sorties et génèrent un niveau logique haut. Les colonnes quant à elles sont placées en entrees et vont donc pouvoir être lues. La donnée émise sur les lignes, qui était initialement 1111, est remplacée par 1110 et la première colonne est lue. Si on y trouve un 0 logique on a identifié la touche actionnée qui se trouve à l'intersection de la première ligne et de la première colonne. Si ce n'est pas le cas, on lit la colonne suivante et ainsi de suite jusqu'à la dernière. Si aucun 0 logique n'a été trouvée, on applique alors 1101 sur les lignes et on recommence le processus précédent. Lorsque la touche a été identifiée on dispose alors d'un code qui correspond à la position de cette dernière sur le clavier. Il ne reste donc plus qu'à le transcoder, au moyen d'un simple tableau, afin de le faire correspondre a l'indication figurant sur la touche, si bien sûr on souhaite disposer de cette information. Dans le cas contraire, ce code " brut " peut être utilisé tel quel pour valider la fonction déclenchée par la touche actionnée. l'écriture en C d'un tel programme reste assez facile grâce à la fonction INPUT et à la fonction OUTPUT que nous verrons dans la suite de cette manipulation, mais la majorité des compilateurs C pour PIC vous epargnent ce travail et sont fournis avec un " driver " ou pilote prêt à l'emploi.
C'est le cas du compilateur C de CCS, que nous utilisons à titre d'exemple, qui propose kbdd.c pour un clavier 12 touches câblees en matrice. Il suffit de l'inclure dans votre programme source pour pouvoir ensuite appeler les fonctions suivantes :

- kbd_init() qui doit être appelée au moins une fois avant toute utilisation de la fonction de lecture du clavier ;
- kbd_getc() qui retourne le caractère correspondant à la touche actionnée, ou un caractère nul dans le cas contraire.

Nous vous laissons le soin d'analyser KBDD.C pour constater que le mode de connexion du clavier est prévu en standard sur le port B ou le port D au choix et que la définition des caractères correspondant aux touches du clavier est paramétrable par vos soins. Nous verrons en effet dans la suite de cette manipulation un exemple d'utilisation conjoint de ce programme avec celui de gestion d'un afficheur à cristaux liquides afin de réaliser un mini terminal.

5.3 Sorties parallèles

Tout aussi utilisées que les entrées parallèles, les sorties parallèles sont mises à contribution pour allumer des LED, actionner des relais ou piloter des afficheurs. Elles sont également simples à mettre en oeuvre car elles ne requièrent elles aussi que très peu de composants externes.

5.3.1 Commande dafficheurs alphanumériques à cristaux liquides

Lorsque l'application à des besoins conversationnels qui ne peuvent plus se satisfaire d'un banal affichage de type chiffres ou symboles simples, on fait tres souvent appel aux affi-

cheurs alphanumériques a cristaux liquides. On trouve en effet aujourd'hui à bas prix sur le marché des modeles autorisant l'affichage d'une, deux ou même quatre lignes de seize à quarante caractères. Qui plus est, ces caractères ne sont plus seulement limités à des chiffres mais à tous les caractères alphanumériques que l'on rencontre sur un clavier de micro-ordinateur. Ces afficheurs sont toujours fournis sous forme d'un petit circuit imprimé support comprenant tout a la fois l'afficheur lui même et l'électronique de gestion. Cette electronique facilite la commande de l'afficheur par le microcontrôleur en le déchargeant de toute la partie gestion de l'afficheur et surtout de la géneration, tout de même assez complexe, des signaux nécessaires.
Les miracles de la " standardisation automatique " dont sont coutumiers les fabricants d'Extreme-Orient font que quasiment tous ces afficheurs utilisent les memes contrôleurs, ou des controleurs compatibles. Ils s'interfacent donc tous avec les mêmes signaux, qui se gerent de la même façon et ils comprennent tous un noyau d'ordres de base communs, même si certains modèles plus évolués que d'autres disposent de quelques commandes ou fonctions complémentaires. Interfacer et gerer un tel afficheur avec un PIC reste une opération très simple comme nous allons le decouvrir maintenant grâce au schéma de la figure 5.2. Les donnees peuvent être codées sur huit lignes appelées DB0 à DB7 mais tous ces afficheurs peuvent aussi fonctionner en mode 4 bits dans lequel les données sont transmises en deux fois sur les seuls 4 bits de poids forts du port B comme c'est le cas figure 5.2.

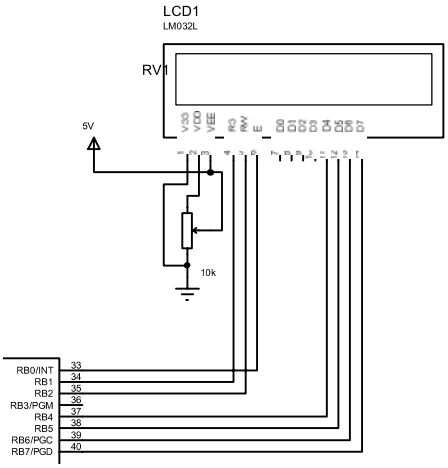

FIG. 5.2 – - Commande directe d'un afficheur alphanumérique à cristaux liquides

Les trois lignes de controle peuvent alors être pilotées par les lignes de poids faibles de ce meme port B. Ces lignes ont les fonctions suivantes, valables pour tous les afficheurs de ce type :
– E ou Enable valide l'afficheur lorsqu'elle est au niveau haut. Il est alors a même de recevoir des commandes ou des données via ses autres lignes. Il reste insensible à leur état dans le cas contraire.
– R/ W pour Read/Write indique si l'on veut écrire une donnee dans l'afficheur ou lire ses informations. Il dispose en effet d'un registre interne capable de fournir certaines

indications d'état.
- RS pour Register Select indique si l'on travaille sur des données (RS = 1) ou sur des commandes (RS = 0).

En mode donnee, l'afficheur affiche successivement les caractères de code ASCII correspondant aux données reçues les uns à la suite des autres, l'avancement de son curseur étant automatique. Un certain nombre de commandes, à voir dans sa fiche technique, permettent une gestion performante de cet affichage : déplacement du curseur de droite à gauche ou l'inverse, avec ou sans effacement de caractere, effacement de tout l'affichage, etc. Ces commandes sont en fait des codes qui doivent être envoyés sur les lignes de donnees de l'afficheur après avoir mis celui-ci en mode commande (RS = 0). Certains modèles disposent même d'une mémoire de génération de caractères interne dans laquelle vous pouvez stocker les formes de caracteres de votre choix. Le dialogue avec un tel afficheur est fort simple et se passe de la façon suivante dans le cas d'une écriture par exemple :
- mise à zéro de la ligne R/W ;
- positionnement de la ligne RS au niveau désiré selon que l'on souhaite envoyer une donnée ou une commande ;
- positionnement du code de la donnée ou de la commande sur DB0 à DB7 ou DB4 à DB7 ;
- mise à un de la ligne E permettant la prise en compte de ces informations ;
- mise à zéro de la ligne E pour rendre a nouveau l'afficheur insensible à l'état de DB0 à DB7.

Ce processus peut se répéter autant de fois que nécessaire mais, compte tenu de la lenteur des afficheurs LCD, un delai d'attente doit être respecté entre l'envoi de deux données ou commandes successives. Sa valeur typique peut aller de 100 us pour un simple affichage de caractère jusqu'à 5 ms pour les operations les plus complexes telles que l'effacement complet de l'afficheur par exemple. Heureusement, une commande de lecture d'état est disponible si nécessaire. La realisation du dialogue avec un tel afficheur ne présente pas de difficultés particulières avec les fonctions INPUT et OUTPUT déjà vues, mais elle se révèle vite assez fastidieuse si l'on veut pouvoir exploiter toutes les possibilites offertes par l'afficheur. Fort heureusement, quasiment tous les compilateurs C pour PIC du marché proposent le " driver " nécessaire en standard et CCS ne fait pas exception à cette règle avec son programme LCD.C. Ce dernier, qu'il suffit d'inclure dans votre programme source, permet ensuite d'appeler les fonctions suivantes :
- lcd_init() qui doit être appelée avant l'utilisation de l'une quelconque des autres fonctions et qui permet d'initialiser l'afficheur ;
- lcd_putc(c) qui affiche le caractere c au prochain emplacement disponible sur l'afficheur mais qui sait aussi interpréter \f (effacement de l'affichage), \n (saut au début de la seconde ligne) et \b (retour arrière du curseur d'une position) ;
- lcd_goto(x, y) qui definit la prochaine position d'affichage utilisée par lcd_putc(c) colonne x, ligne y, sachant que le premier caractère de la première ligne est repéré 1,1 ;
- lcd_getc(x, y) qui renvoie le caractere affiche en colonne x de la ligne y avec les mêmes conventions que ci-dessus.

Tel qu'il est fourni, ce programme est prevu pour un afficheur câblé comme indiqué figure 5.2 sur le port D ou sur le port B selon que l'on laisse en commentaire ou non sa ligne :

#define use_portb_lcd TRUE
De plus, le mode de connexion aux lignes du port est également par défaut celui visible figure 2 mais peut être largement adapté à vos besoins en modifiant la structure définie au début du programme et reproduite ci-dessous :

```
struct lcd_pin_map {
                    BOOLEAN enable;     // Ligne afficheur connecté sur XO
                    BOOLEAN rs;         // Ligne afficheur connecté sur X1
                    BOOLEAN rw ;        // Ligne afficheur connecté sur X2
                    BOOLEAN unused;     // Ligne afficheur connecté sur X3
                    int data : 4 ;       // DB4 à DB7 sur X4 à X7
                    } lcd
```

Les lignes de donnees de l'afficheur doivent rester sur X4 à X7 (où X est égal à B ou D selon le port choisi). Par contre, les lignes de controle peuvent être réparties comme bon vous semble sur les lignes de port restantes comme indiqué ci-dessus (unused correspond à la seule ligne de port qui reste inutilisée). Ainsi par exemple, si vous voulez modifier ce " driver " pour l'adapter au mode de cablage de l'afficheur LCD, il vous faudra modifier la structure ci-dessus de la façon suivante :

```
struct lcd_pin_map {
                    BOOLEAN unused      // Ligne afficheur connecté sur BO
                    BOOLEAN rw;         // Ligne afficheur connecté sur Bl
                    BOOLEAN rs;         // Ligne afficheur connecté sur B2
                    BOOLEAN enable;     // Ligne afficheur connecté sur B3
                    int data : 4;       // DB4 à DB7 sur B4 à B7
                    } lcd ;
```

5.3.2 Entrées/sorties parallèles (simultanées)

Les lignes d'entrées/sorties parallèles du PIC pouvant changer de sens de fonctionnement par programme, et donc aussi souvent qu'on le souhaite, il est intéressant d'exploiter cette possibilité pour permettre l'interfaçage d'un organe d'entrée et d'un organe de sortie sur le même port. Le recours le plus fréquent à ce mode d'exploitation particulier a lieu lorsque l'application doit dialoguer avec un opérateur humain au moyen d'un clavier et d'un afficheur. En effet, la vitesse de reaction d'un être humain est sans commune mesure avec celle d'un PIC qui peut donc tour a tour lire un clavier et piloter un afficheur sur le même port, sans que nous ne nous apercevions de quoi que ce soit et surtout sans risquer de perdre la moindre saisie sur le clavier. La figure 5.3 montre un exemple de câblage d'une telle configuration mettant en oeuvre un clavier à 12 touches, style clavier télephonique, câble en matrice 4x3, et un afficheur alphanumérique à cristaux liquides tel celui que nous venons d'etudier ci-dessus. Grâce aux programmes de gestion de ces deux

éléments fournis par CCS, l'écriture du programme global nécessaire à notre application devient fort simple comme le montre à titre d'exemple le listing 6 suivant.

```
\\***************************Listing6*************************************\\
#include <16F877.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay(clock=4000000)
#include <lcd.c>
#include <kbd.c>
void main() {
       char k;
       lcd_init();               // Initialisation de l'afficheur
       kbd_init();               // Initialisation du clavier
       lcd_putc("\fPrêt...\n");  // Affichage "prêt"
       while (TRUE) {            // Boucle sans fin
              k=kbd_getc();          // Lecture clavier
              if(k!=0)
              if(k=='*')                  // Si touche *
                     lcd_putc('\f');      // Effacement de l'afficheur else
              lcd_putc(k);                // Sinon affichage touche
              }
       }
\\************************************************************\\
```

Cet exemple se contente de lire le clavier et d'afficher les touches frappees sur l'afficheur, sachant que la touche * realise un effacement de ce dernier et un positionnement du curseur au début de la première ligne. L'extrapolation de ce programme vers des fonctions plus ambitieuses ne présente evidemment aucune difficulté en utilisant ce canevas de départ.

5.4 Travail demandé

1. Compiler le programme avec PCW par la suite vérifier son bon fonctionnement à l'aide de la simulation par ISIS.
2. Expliquer puis commenter le programme présenté dans les deux listing 6.
3. Modifier le programme pour afficher votre noms lorsque vous tapez la touche '1'.
4. Programmer le microcontrôleur sur la carte d'expérimentation et verifier le fonctionnement de votre programme.

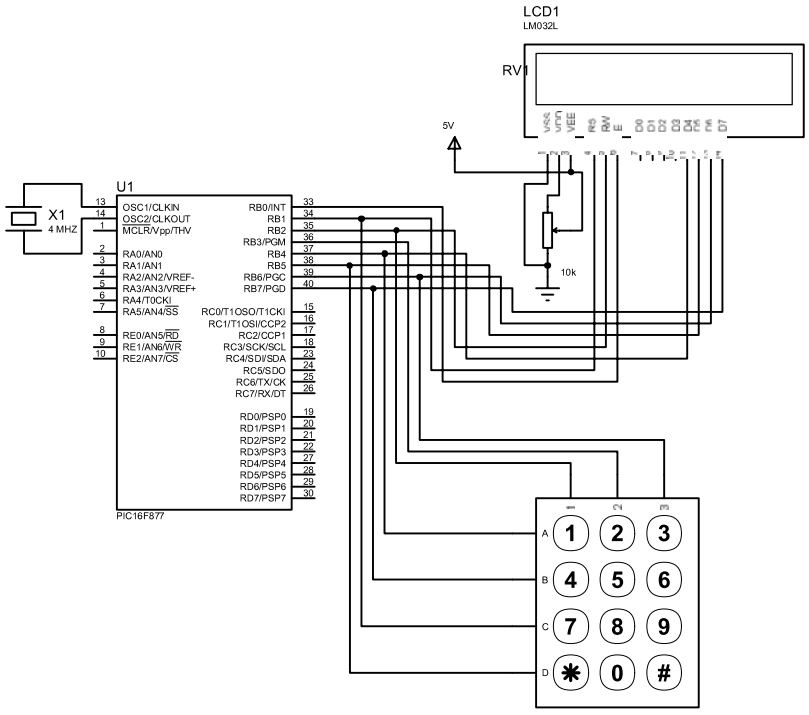

FIG. 5.3 – Utilisation des mêmes lignes d'un port parallèle pour lire un clavier en matrice et commander un afficheur LCD