Commande d'afficheurs alphanumériques à cristaux liquides

Lorsque l'application a des besoins conversationnels qui ne peuvent plus se satisfaire d'un banal affichage de type chiffres ou symboles simples, on fait très souvent appel aux afficheurs alphanumériques à cristaux liquides. On trouve en effet aujourd'hui à bas prix sur le marché des modèles 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 à la fois l'afficheur lui même et l'électronique de gestion. Cette électronique facilite la commande de l'afficheur par le microcontrôleur en le déchargeant de toute la partie gestion de l'afficheur et surtout de la génération, tout de même assez complexe, des signaux nécessaires.
Les miracles de la « standardisation automatique » font que quasiment tous ces afficheurs utilisent les mêmes contrôleurs, ou des contrôleurs compatibles. Ils s'interfacent donc tous avec les mêmes signaux, qui se gèrent 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 gérer un tel afficheur avec un PIC reste une opération très simple comme nous allons le découvrir maintenant grâce au schéma de la figure 7.13.
Les données 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 7.13. Les trois lignes de contrôle peuvent alors être pilotées par les lignes de poids bibles de ce même port B. Ces lignes ont les fonctions suivantes, valables pour tous les afficheurs de ce type:

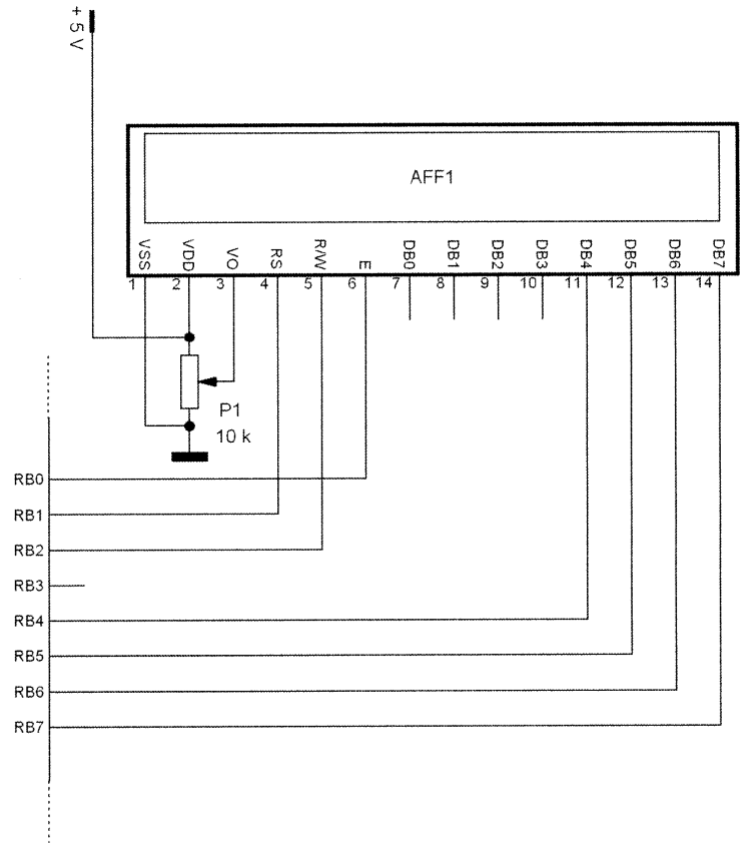


Figure 7.13 – Commande directe d'un afficheur alphanumérique à cristaux liquides.

E ou Enable valide l'afficheur lorsqu'elle est au niveau haut. Il est alors à même de recevoir des commandes ou des données via ses autres lignes. Il reste insensible à leur état dans le cas contraire.
$R/\overline{W}$ pour $Read/\overline{Write}$ indique si l'on veut écrire une donnée 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).

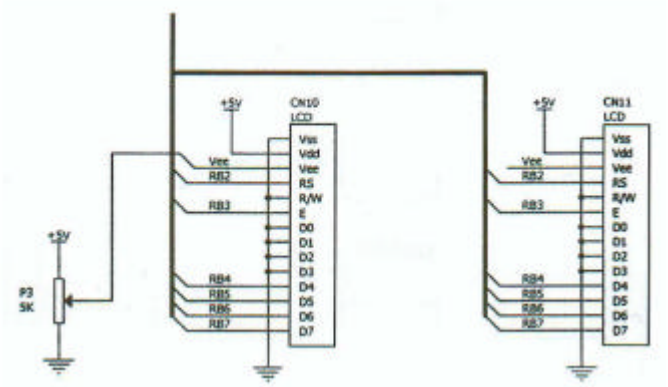
En mode données, l'afficheur affiche successivement les caractères de code ASCII correspondant aux données reçues les uns à la suite des autres, l'avancement de son curseur étant automatique. Un certain nombre de commandes, à voir dans sa fiche technique, permettent une gestion performante de cet affichage: déplacement du curseur de droite à gauche ou l'inverse, avec ou sans effacement de caractère, effacement de tout l'affichage, etc. Ces commandes sont en fait des codes qui doivent être envoyés sur les lignes de données 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 caractères 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:
1=> mise à zéro de la ligne R/W ;
2=> positionnement de la ligne RS au niveau désiré selon que l'on souhaite envoyer une donnée ou une commande;
3=> positionnement du code de la donnée ou de la commande sur DB0 à DB7 ou DB4 à DB7;
4=> mise à un de la ligne E permettant la prise en compte de ces informations;
5=> mise à zéro de la ligne E pour rendre à 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 délai d'attente doit être respecté entre l'envoi de deux données ou commandes successives. Sa valeur typique peut aller de 100 µs pour un simple affichage de caractère jusqu'à 5 ms pour les opérations 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 réalisation 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 possibilités 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 _ini t () 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 caractère 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 définit la prochaine position d'affichage utilisée par lcdputc (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 caractère affiché en colonne x de la ligne y avec les mêmes conventions que ci-dessus.

Tel qu'il est fourni, ce programme est prévu pour un afficheur câblé comme indiqué figure 7.13 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 7.13 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 connectée sur X0
        BOOLEAN     rs;                     // Ligne afficheur connectée sur X1
        BOOLEAN     rw;                     // Ligne afficheur connectée sur X2
        BOOLEAN     unused;                 // Ligne afficheur connectée sur X3
        Int         data : 4;               // DB4 à DB7 sur X4 à X7
} lcd ;
```

Les lignes de données 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 contrôle peuvent être réparties comme bon vous semble sur les lignes de port restantes du port B ou C choisi 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 câblage de l'afficheur LCD prévu sur la carte présentée ci dessous, il vous faudra modifier la structure ci-dessus de la façon suivante:




```
Struct lcd_pin_map
{
        BOOLEAN     unused;         // Ligne afficheur connectée sur B0
        BOOLEAN     rw;                     // Ligne afficheur connectée sur B1
        BOOLEAN     rs;                     // Ligne afficheur connectée sur B2
        BOOLEAN     enable;                 // Ligne afficheur connectée sur B3
        Int         data : 4;               // DB4 à DB7 sur B4 à B7
} lcd ;
```

Remarquez à ce propos que nous avons fait figurer R/W comme étant reliée à B1 dans cette structure alors que cette ligne est mise à la masse sur la carte Easy PIC 2. Nous avons en effet estimé judicieux de modifier cette carte de façon à pouvoir commander la ligne R/W de l'afficheur au moyen d'une des lignes du port B, en l'occurrence B1. Il suffit pour cela de couper la patte 5 du connecteur destiné à l'afficheur et de relier le plot 5 de ce dernier à la patte BO du support à 40 pattes du PIC.

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 réaction d'un être humain est sans commune mesure avec celle d'un PIC qui peut donc tour à 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 7.14 montre un exemple de câblage d'une telle configuration mettant en œuvre un clavier à 12 touches, style clavier téléphonique, câblé en matrice 4 X 3, et un afficheur alphanumérique à cristaux liquides tel celui que nous venons d'étudier 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 7.2.
Listing 7.2

```
#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("\fPret...\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 frappées sur l' afficheur, sachant que la touche * réalise 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 évidemment aucune difficulté en utilisant ce canevas de départ.

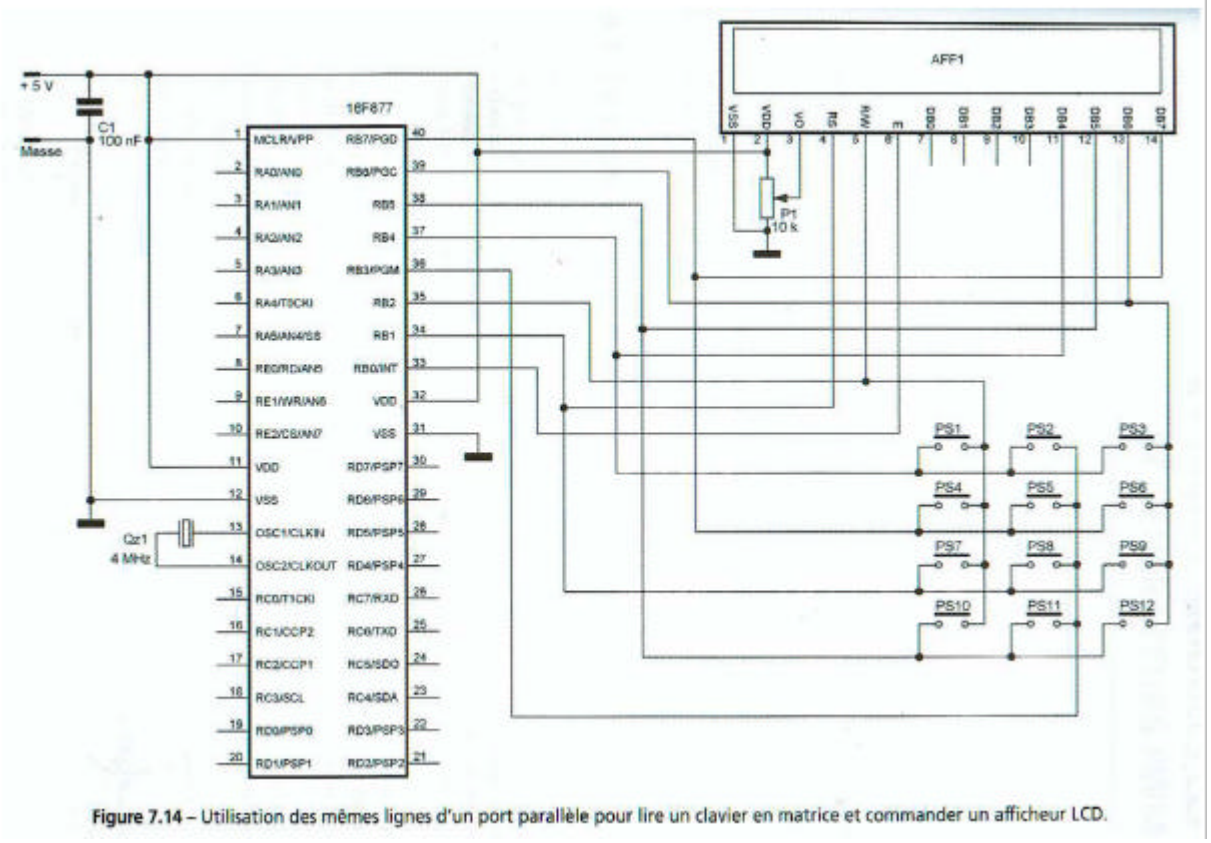


Figure 7.14 – Utilisation des mêmes lignes d'un port parallèle pour lire un clavier en matrice et commander un afficheur LCD.

Résumé de la documentation de l'afficheur :

Instruction	Rs	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Fonction	Temps
(1) Clear_display()	0	0	0	0	0	0	0	0	0	1	Nettoie tout l'afficheur et place le curseur en début de ligne (adresse 0).	1,64 ms
(2) Return_home()	0	0	0	0	0	0	0	0	1	x	Positionne le curseur au début de la ligne sans modification du texte affiché. La DD Ram est inchangée.	1,64 ms
(3) Entry_mode() Set	0	0	0	0	0	0	0	1	I/D	S	Force le mouvement du curseur et spécifie si l'affichage sera décalé ou pas quand les données seront lues ou écrites. si I/D et S prennent les états logiques: 0 0 => Le texte affiché est fixe, le curseur défile vers la gauche avec insertion caractères. 0 1 => Le curseur est fixe, le texte affiché défile vers la droite avec insertion caractères. 1 0 => Le texte affiché est fixe, le curseur défile vers la droite avec insertion caractères. 1 1 => Le curseur est fixe, le texte affiché défile vers la gauche avec insertion caractères.	40 µs
(4) Display() ON/OFF controle	0	0	0	0	0	0	1	D	C	B	Valide ou non (ON/OFF) l'affichage (D) et le curseur (C) et indique si le curseur sera clignotant ou non (B). D: 0 =>extinction afficheur 1 =>allumage afficheur C: 0 =>pas de curseur 1 =>présence curseur B: 0 =>curseur fixe 1 =>curseur clignotant	40 µs
(5) Cursor_Display_shift()	0	0	0	0	0	1	S/C	R/L	x	x	Déplace le curseur et décale l'affichage sans modification de la DD RAM. Si S/C et R/L prennent les états logiques: 0 0 => Déplace le curseur de 1 vers la gauche, le texte affiché n'est pas modifié. 0 1 => Déplace le curseur de 1 vers la droite, le texte affiché n'est pas modifié. 1 0 => Déplace le texte affiché de 1 vers la gauche, le curseur suit. 1 1 => Déplace le texte affiché de 1 vers la droite, le curseur suit.	40 µs
(6) Function set	0	0	0	0	1	DL	N	F	x	x	Définit le nombre de bits de donnée(DL), le nombre de ligne(N) et la fonte des caractères(F). DL=0 => 4 bits de donnée DL=1 => 8 bits de données N =0 => affichage sur 1 ligne N =1 => affichage sur 2 lignes F =0 => caractères de 5*7 pixels F =1=> carcatères de 5*10 pixels	40 µs
(7) CG Ram address set	0	0	0	1			A	C	G		Définit l'adresse A_{CG} de la CG RAM qui démarre un transmission ou une réception de donnée CG RAM.	40 µs
(8) DD Ram address set	0	0	1				A	D	D		Définit l'adresse A_{DD} de la DD RAM qui démarre un transmission ou une réception de donnée DD RAM.	40 µs
(9) BF/S address read	0	1	BF				A	C			Lit le drapeau BF indiquant que l'opération interne s'accomplie est lit le contenu du compteur d'adresse AC. (utilisé pour la CG RAM ou la DD RAM).	1 µs
(10) Data write to CG Ram or DD Ram	1	0									Ecrit une donnée dans la DDRAM ou dans la CGRAM.	40 µs
(11) Dat read from CG Ram or DD Ram	1	1									Lit une donnée depuis la DDRAM ou depuis la CGRAM.	40 µs