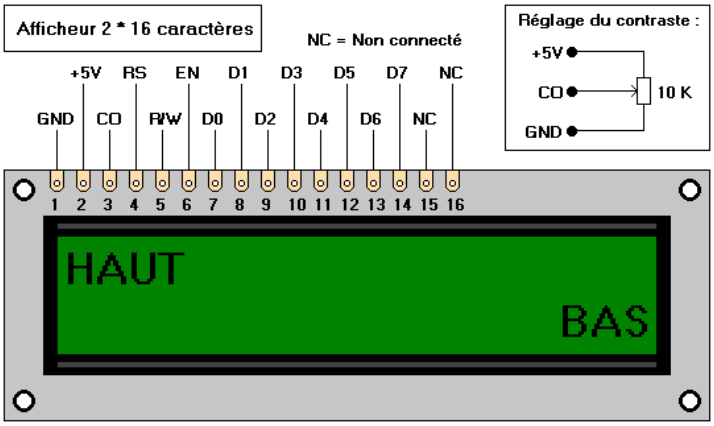
Introduction aux Systèmes embarqués

CHAP 3 : Gestion des afficheurs LCD

1. Introduction :

Les afficheurs à cristaux liquides, autrement appelés afficheurs LCD (Liquid Crystal Display), sont des modules compacts intelligents et nécessitent peu de composants externes pour un bon fonctionnement. Ils consomment relativement peu (de 1 à 5 mA), sont relativement bons marchés et s'utilisent avec beaucoup de facilité.



Plusieurs afficheurs sont disponibles sur le marché et diffèrent les uns des autres, non seulement par leurs dimensions, (de 1 à 4 lignes de 6 à 80 caractères), mais aussi par leurs caractéristiques techniques et leur tension de service. Certains sont dotés d'un rétro-éclairage de l'affichage. Cette fonction fait appel à des LED montées derrière l'écran du module, cependant, cet éclairage est gourmand en intensité (de 80 à 250 mA).

2. Brochage :

Un circuit intégré spécialisé est chargé de la gestion du module. Il remplit une double fonction : d'une part il commande l'affichage et de l'autre se charge de la communication avec l'extérieur.

No	Nom	Rôle de la broche
1	Masse	Alimentation : masse 0 V de l'afficheur
2	Vcc	Alimentation : +5 V de l'afficheur
3	Vo	Réglage de contraste entre 0 et +5 V (plus proche de la masse)
4	RS	Commutation du registre entre les instructions « 0 » et les données « 1 »
5	R/W	Commutation entre lecture « 1 » (Read) et écriture « 0 » (Write)
6	E	Entrée de validation (activation sur le front descendant de l'impulsion positive)
7	D0	Bus de données à trois états : « 0 », « 1 », ou haute impédance
8	D1	
9	D2	
10	D3	
11	D4	
12	D5	
13	D6	
14	D7	
15	BI+	Anode du rétro éclairage (+5 V)
16	BI-	Cathode du rétro éclairage (masse)

L'Arduino dispose d'une librairie *LiquidCrystal* qui permet de contrôler un afficheur LCD alphanumérique standard basé sur le circuit intégré Hitachi HD44780 (ou compatible), ce qui est le cas pour la plupart des afficheurs alphanumériques LCD disponibles.

3. LiquidCrystal (Constructeur de classe) :

Cette librairie permet de créer une variable de type LiquidCrystal en définissant les broches utilisées avec le LCD et son mode de fonctionnement. L'afficheur LCD peut être contrôlé avec 4 ou 8 lignes de données. Si vous souhaitez utiliser l'afficheur en mode 4 bits, omettez les numéros des broches pour d0 à d3 et laissez ces broches non-connectées. La broche RW peut-être connectée à la masse (0V) au lieu d'être connectée à une broche de la carte Arduino. De la même façon, si vous ne souhaitez pas utiliser cette broche, omettez-là dans les paramètres de la fonction LiquidCrystal() (voir ci-dessous).

Note : Un afficheur LCD peut être contrôlé uniquement avec 4 broches de données (d4 à d7) et 2 broches de commandes (RS et E), soit 6 broches seulement, contre 8 broches de données et 3 broches de commandes en connexion complète. On économise donc ainsi 5 broches de la carte Arduino qui resteront ainsi disponibles.

Plusieurs formes sont possibles : les broches omises déterminent le mode de fonctionnement utilisé.

```
LiquidCrystal lcd(rs, enable, d4, d5, d6, d7); // mode 4 bits – RW non connectée
LiquidCrystal lcd(rs, rw, enable, d4, d5, d6, d7); // mode 4 bits – RW utilisée
LiquidCrystal lcd(rs, enable, d0, d1, d2, d3, d4, d5, d6, d7); // mode 8 bits –
//RW non connectée
LiquidCrystal lcd(rs, rw, enable, d0, d1, d2, d3, d4, d5, d6, d7); /*mode 8 bits
//– RW utilisée */
```

- lcd: le nom de la variable LiquidCrystal déclarée.
- rs: le numéro de la broche de l'Arduino qui est connecté à la broche RS de l'afficheur LCD.
- rw: le numéro de la broche de l'Arduino qui est connecté à la broche RW de l'afficheur.
- enable: le numéro de la broche de l'Arduino qui est connecté à la broche E(enable) de l'afficheur.
- d0, d1, d2, d3, d4, d5, d6, d7 : le numéro de la broche de l'Arduino qui est connecté aux broches correspondantes des broches de données de l'afficheur LCD. Les broches de données d0, d1, d2, and d3 sont optionnelles ; si elles sont omises, le LCD pourra être contrôlé en utilisant seulement les quatre broches de données d4, d5, d6, d7 (mode 4 bits).

Exemple :

```
#include<LiquidCrystal.h>
LiquidCrystal lcd(11, 10, 5, 4, 3, 2); // déclare une variable LiquidCrystal
//appelée lcd
// mode 4 bits et RW pas utilisé
void setup()
{
 lcd.print("hello, world!"); // affiche le désormais célèbre texte "Hello World!"
}
void loop() {}
```

4. Les fonctions disponibles

4.1. Fonctions d'initialisation

- **begin() :** Spécifie les dimensions de l'afficheur LCD (nombre de colonnes et nombre de lignes) (intégrée dans la fonction setup().

```
lcd.begin(colonnes, lignes)
```

- lcd: une variable de type LiquidCrystal.
- colonnes: le nombre de colonnes de l'afficheur LCD utilisé.

- lignes: le nombre de lignes de l'afficheur LCD utilisé.

4.2. Fonctions d'écriture :

- **print() :** Affiche un texte ou un nombre sur l'afficheur LCD.

```
lcd.print("texte")
lcd.print(data)
lcd.print(data, BASE)
```

- lcd : une variable de type LiquidCrystal.
- "texte" : une chaîne de caractères
- data : la donnée à afficher (char, byte, int, long, float ou string)
- BASE (optionel): la base dans laquelle vous voulez afficher les nombres : BIN pour binaire (base 2), DEC pour décimal (base 10 - base par défaut), OCT pour octal (base 8) et HEX pour hexadécimal (base 16).

- **write()** :Ecrit un caractère sur l'afficheur LCD.

```
lcd.write(data)
```

- lcd: une variable de type LiquidCrystal
- data: le caractère à écrire sur l'écran

4.3. Fonctions de gestion de l'écran

- **clear() :** Efface l'écran et positionne le curseur dans le coin supérieur gauche de l'écran.

```
lcd.clear()
```

- **display()** :Réallume l'écran de l'afficheur LCD , après qu'il ait été éteint avec l'instruction noDisplay(). Ceci réactive à l'identique le texte et le curseur de l'affichage tels qu'ils étaient.

```
lcd.display()
```

- **noDisplay()** : Eteint l'écran de l'afficheur LCD, sans perdre le texte actuellement affiché (autrement dit, l'afficheur lui-même n'est pas mis hors tension, seul l'écran est éteint).

```
lcd.noDisplay()
```

4.4. Fonctions de positionnement du curseur :

- **home() :** Positionne le curseur dans le coin gauche de l'écran LCD. Le texte sera dorénavant écrit à partir de cette position. Pour effacer également l'écran, utiliser plutôt l'instruction clear().

```
lcd.home()
```

- **clear() :** Efface l'écran et positionne le curseur dans le coin supérieur gauche de l'écran.

```
lcd.clear()
```

- **setCursor() :** Positionne le curseur de l'afficheur LCD à la localisation voulue (colonne,ligne), et donc définit la position à laquelle le texte sera dorénavant affiché à l'écran.

```
lcd.setCursor(colonne, ligne)
```

- lcd: une variable de type LiquidCrystal
- colonne: la colonne à laquelle positionner le curseur (ATTENTION : 0 est la 1ère colonne…).
- ligne: la ligne à laquelle positionner le curseur (ATTENTION : 0 est la 1ère ligne…).

4.5. Fonctions modifiant l'aspect du curseur :

- **cursor() :** Affiche le curseur sous forme d'un trait de base (_) sous l'emplacement courant du curseur, là où sera écrit le prochain caractère.

```
lcd.cursor()
```

- **noCursor()** : Masque le curseur (trait de base sous l'emplacement courant).

```
lcd.noCursor()
```

- **blink() :** Affiche le clignotement du curseur à la position courante.

```
lcd.blink()
```

- **noBlink()** :Désactive le clignotement du curseur.

```
lcd.noBlink()
```

4.6. Fonctions de contrôle du comportement du curseur :

- **autoscroll() :** Active le décalage automatique de l'afficheur LCD. Ceci entraîne un décalage de 1 espace des caractères précédents déjà affichés par chaque nouveau caractère. Si la direction courante est de gauche à droite (par défaut), l'écran décale vers la gauche; si la direction courante est de droite à gauche, l'afficheur décale vers la droite. Ceci a pour effet de sortir chaque nouveau caractère à la même localisation sur l'afficheur LCD. Autrement dit, tout se passe comme si le curseur reste fixe : c'est le texte qui décale.

```
lcd.autoscroll()
```

- **noAutoscroll() :** Désactive le décalage automatique du LCD.

```
lcd.noAutoscroll()
```

- **leftToRight() :** Initialise la direction d'écriture du texte sur l'écran LCD de gauche à droite (valeur par défaut au démarrage). Ceci signifie que les caractères suivants écrits sur l'écran iront de gauche à droite, mais cela n'affectera pas le texte précédent déjà affiché.

```
lcd.leftToRight()
```

- **rightToLeft() :** Initialise la direction d'écriture du texte sur l'afficheur LCD de droite à gauche (la direction par défaut est de gauche à droite). Ceci signifie que le texte écrit à la suite des caractères déjà affichés sur l'écran sera de droite à gauche, mais cela n'affectera pas le texte précédemment affiché.

```
lcd.rightToLeft()
```

4.7. Fonctions d'effets visuels :

- **scrollDisplayLeft() :** Décale le contenu de l'écran (texte et curseur) d'un espace vers la gauche.

```
lcd.scrollDisplayLeft()
```

- **scrollDisplayRight() :** Décale le contenu de l'écran (texte et cursuer) d'un espace vers la droite.

```
lcd.scrollDisplayRight()
```

4.8. Fonction de création de caractère personnalisé :

- **createChar() :** Crée un caractère personnalisé pour l'utiliser sur le LCD. Jusqu'à 8 caractères de 8 pixels sont supportés (numérotés de 0 à 7. L'apparence de chaque caractère personnalisé est définie par un tableau de 8 octets, un pour chaque ligne. Les 5 bits de poids faible de chaque

octet défini les pixels affichés dans chaque ligne, Pour afficher un caractère personnalisé sur l'écran, utiliser l'instruction write (numéro) où numéro est le numéro du caractère (de 0 à 7).

```
createChar(numero, tableau)
```

- Numero : le numéro du caractère créé
- Tableau : tableau de 8 octets dont les 5 octets de poids faible définissent pixels affichés ((1 octet pour chaque ligne).