LES AFFICHEURS À CRISTAUX LIQUIDES

COMMANDE D'AFFICHEURS À CRISTAUX LIQUIDES

1) INTRODUCTION

Ce cours présente :
- l'interfaçage matériel d'afficheurs à cristaux liquides à un µP/µC
- la commande logicielle d'afficheurs à cristaux liquides

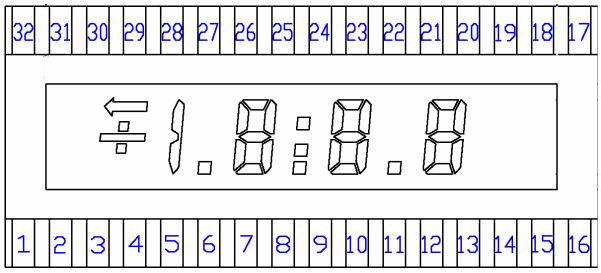
Les afficheurs à cristaux liquides (LCD = Liquid Crystal Display) peuvent être avec ou sans contrôleur intégré. Lorsqu'un contrôleur est intégré, on parle de module LCD.

L'interfaçage peut être :
- direct avec certains afficheurs LCD sur des versions de µC prévues pour cet usage
- direct avec les afficheurs avec contrôleur intégré
- à travers un circuit spécialisé pour les afficheurs sans contrôleur intégré (afficheurs à segments, etc.)

Un § est consacré à chaque type de LCD et d'interfaçage.
Pour certaines commandes, il est utile de connaître quelques notions sur le fonctionnement des afficheurs LCD. La présentation est effectuée avant les interfaçages directs ou à travers un circuit spécialisé.

Il existe de très nombreux types de LCD.
Un afficheur peut être constitué de :
- segments (affichage de chiffres ou lettres)
- pictogrammes, dessins, icônes
- mots ou groupes de mots (ex : Low Batt)
- points (affichage alphanumérique ou graphique)

La figure ci-contre représente un afficheur, sans contrôleur intégré, constitué de segments, point et un pictogramme (flèche).



2) AFFICHEURS 2 LIGNES DE 16 CARACTÈRES OU 4 LIGNES DE 20 CARACTÈRES AVEC CONTRÔLEUR INTÉGRÉ

Il existe de nombreux types d'afficheurs avec contrôleur intégré. Les plus répandus sont ceux avec **2 lignes de 16 caractères** intégrant un contrôleur Hitachi HD44780 ou équivalent (KS0066 de Samsung, etc.). Ils sont quasiment normalisés au niveau des broches et des commandes.

Les afficheurs de 4 lignes de 20 caractères sont assez répandus. Ils sont eux aussi quasiment normalisés au niveau des broches et des commandes.
Seuls les afficheurs de ces types sont décrits ici.

D'autres afficheurs à 1 ligne se commandent de façon similaire. Il est donc facile de transposer ce qui est mentionné ici.

2.1) PRÉSENTATION DE L'AFFICHEUR

Le type d'afficheur étudié permet d'afficher :

- tous les caractères ASCII internationaux (codes 20h à 7Dh ou 7Eh) : chiffres et lettres, caractères de ponctuations, opérateurs arithmétiques, etc.
- les caractères nationaux (à partir des codes 80h ou A0h). Le jeu de caractère dépend de la version du contrôleur
- des caractères définis par l'utilisateur (jusqu'à 8 caractères définis par l'utilisateur)

L'afficheur permet aussi de placer un curseur ou de faire clignoter le caractère à l'endroit ou le prochain caractère doit être affiché. On peut réaliser des animations en décalant la fenêtre d'affichage.

L'afficheur est constitué de :

- un CI contrôleur Hitachi HD44780 ou équivalent
- un ou plusieurs CI unité d'extension (le contrôleur seul ne permet que de piloter 2 lignes de 8 caractères) :
  - un CI pour 2 lignes de 16 caractères
  - plusieurs CI pour 4 lignes de 20 caractères
- une matrice LCD de 2 lignes de 16 caractères ou de 4 lignes de 40 caractères

Le module dispose de :

- signaux de liaison avec le µC ou le système à µP de commande : DB7-DB0, EN, R/W, RS
- 2 broches pour l'alimentation VDD et VSS (VDD – VSS = 5V)
- une broche pour régler de contraste de l'affichage (V0)
- parfois deux broches pour l'alimentation de DELs pour le rétro-éclairage

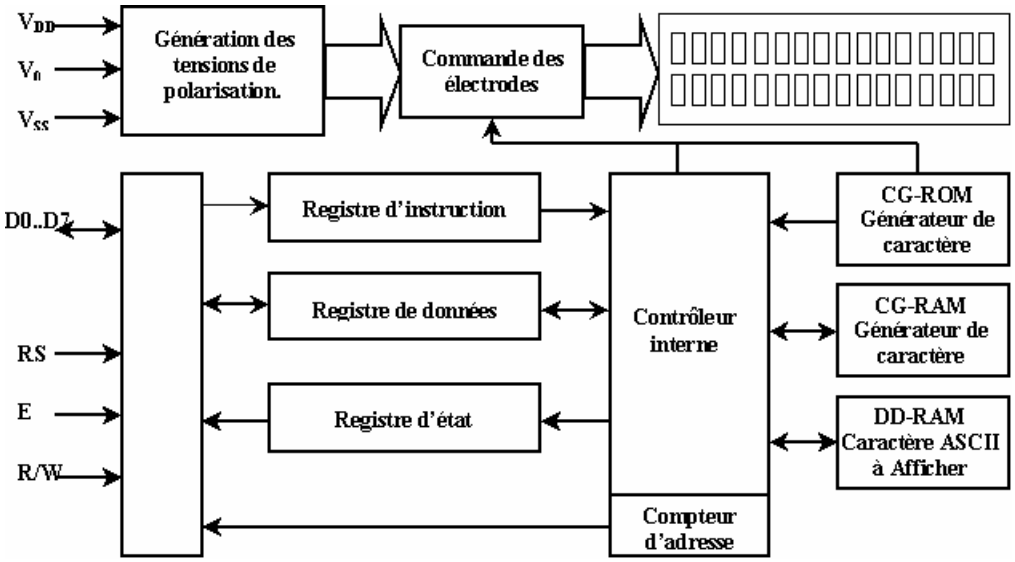
Brochage :

broche **N°**	**Symbole**	**Fonction**
1	VSS	0V
2	VDD	5V
3	V0	Réglage contraste[1]
4	RS	0 : Code instruction 1 : Donnee
5	R/W	1 : Lecture 0 : Ecriture
6	E	Horloge
7	D0	
8	D1	
9	D2	
10	D3	Bus de donnée Bidirectionnel[2]
11	D4	
12	D5	
13	D6	
14	D7	

1 : le réglage de contraste s'effectue en appliquant une tension variable sur cette broche.

2 : Lors de l'utilisation en mode 4 bits seul les bits D4 à D7 sont utilisés. Les lignes correspondant de D0 à D4 sont laissées en l'air. On communique le quarté de poids fort puis de poids faible.

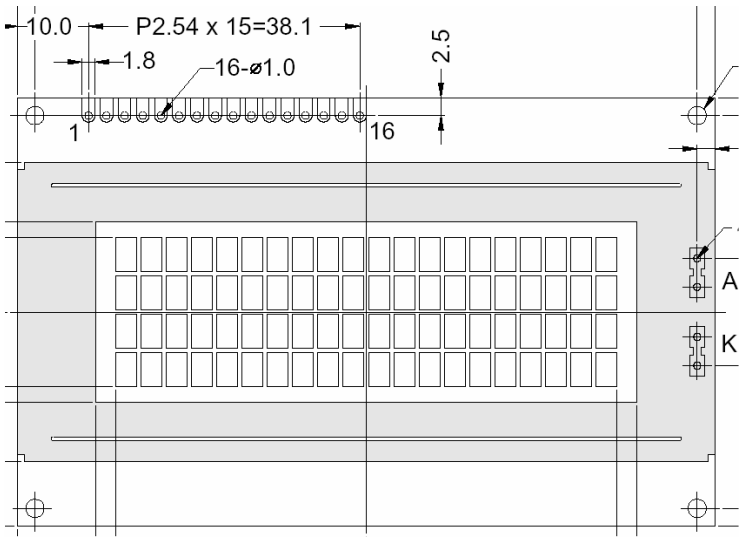
La structure interne très simplifiée d'un afficheur de 2 lignes de 16 caractères est donnée sur la figure suivante[1] :



Le CI contrôleur et le circuit d'extension réalisent toutes les fonctions représentées ci-dessus, hormis l'affichage proprement dit.

Le brochage est quasi normalisé, avec l'espacement entre les pastilles. Cependant l'emplacement des pastilles par rapport au bord de la carte varie d'un afficheur à l'autre.

Ci-contre, un exemple de module :



Pour une bonne utilisation, il est nécessaire de comprendre comment fonctionne le contrôleur HD44780.
Le multiplexage de la matrice LCD est transparent à l'utilisateur, aussi il n'en est pas fait mention ici.

Le rôle des signaux d'interfaçage est détaillé après des généralités sur le contrôleur d'afficheur.

[1] Ce schéma est extrait d'un document de Giampiero D'AQUINO disponible sur Internet

2.3) PRINCIPE SIMPLIFIÉ DE FONCTIONNEMENT DU CONTRÔLEUR

2.3.1) INITIALISATION

Le contrôleur doit d'abord être initialisé (nombre de lignes de l'afficheur, nb de bits de données à utiliser pour communiquer avec le µC, ...). Cette initialisation est nécessaire car le constructeur a prévu de pouvoir adapter son circuit à plusieurs types d'afficheurs et de µP/µC.
L'initialisation s'effectue normalement automatiquement à la mise sous tension avec un circuit interne. Comme ce circuit ne fonctionne pas toujours (la pratique montre qu'il ne fonctionne jamais), il est nécessaire de prévoir systématiquement une séquence d'initialisation commandée par le µC (voir plus loin).

C'est lors de l'initialisation qu'est notamment fixée la largeur du bus de données utilisé pour communiquer avec le µP/µC.
Le module peut communiquer avec le µP/µC par un bus de données de 4 bits ou de 8 bits. Lorsqu'un bus de 4 bits est utilisé, il faut 2 accès successifs au lieu d'un avec un bus de 8 bits.

Les échanges avec un bus de 4 bits permettent de réduire le nombre de broches nécessaires sur le µP/µC.

2.3.2) FONCTIONNEMENT APRÈS INITIALISATION

L'utilisateur écrit dans une RAM, Display Data RAM –**DDRAM**-, les codes des caractères (chacun sur un octet) à afficher et donne des instructions pour l'affichage.

Un générateur de caractère permet de délivrer les informations pour la commande des pixels à partir du code du caractère en DDRAM.

Selon le code du caractère, le circuit utilise :
- un générateur de caractère en ROM (pour la plupart des codes)
- ou un générateur en RAM (pour les codes 00h à 0Fh).

Dans ce dernier cas, l'utilisateur doit au préalable charger la RAM du générateur de caractère, Character Generator RAM –**CGRAM**-, avec les modèles qu'il a défini.

Vu du côté utilisateur, la DDRAM et la CGRAM correspondent à des zones de RAM avec des adresses externes définies.

Les caractères en ROM correspondent aux caractères ASCII pour les codes 21h à 7Dh (lettres minuscules non accentuées, lettres majuscules, chiffres, caractères de ponctuation, signes mathématiques, ...). Pour les autres codes, cela dépend de la version du contrôleur. Voir doc du module afficheur.

Les codes des caractères fournis par l'utilisateur sont rangés dans la DDRAM aux adresses délivrées par un **compteur d'adressage** ou compteur d'adresse.
Le contenu de ce compteur peut être :
- fixé par l'utilisateur à une valeur définie avec une instruction spécifique
- ou remis à 0 avec une instruction qui replace également l'affichage dans sa position d'origine.

Après chaque écriture d'un code de caractère, le compteur est automatiquement incrémenté (ou décrémenté). Ceci permet de simplifier l'écriture du programme utilisateur.
Les modèles du générateur de caractère sont rangés dans la CGRAM aux adresses fixées par le même compteur d'adressage utilisé pour la DDRAM. Le contenu de ce compteur peut être fixé par l'utilisateur. Après chaque écriture, le compteur est automatiquement incrémenté (ou décrémenté).

Le système à µP ou le µC doit fournir au module :
- **des informations pour l'initialisation**
- **des données à placer en DDRAM ou CGRAM**
- **l'adresse de départ pour le rangement des données précédentes**
- **des instructions pour l'affichage**

En retour le module fournit au µC les informations suivantes:
- l'état du module (traitement en cours d'une instruction ou prêt à recevoir une nouvelle instruction ou donnée)
- les données enregistrées dans les RAM
- l'adresse du compteur d'adressage

Ces informations sont souvent inexploitées ou peu exploitées.

Pour simplifier les échanges entre le module et le µC, seuls 2 registres sont utilisés pour les transferts au niveau du module :
- le registre d'instruction **IR** (Instruction Register). Lors d'une écriture par le µC, il contient les codes des instructions et les adresses pour les RAM. Lors d'une lecture par le µC, il contient le bit d'état et l'adresse du compteur d'adresse.
- le registre de données **DR** (Data Register). Il sert à mémoriser les données à écrire (ou à lire) dans (depuis) une des RAM.

Les transferts internes au contrôleur sont transparent à l'utilisateur et dépendent de la succession des commandes (instruction, écriture données) envoyées au modules.

Le bit **RS** (Register Select) positionné par le µC permet d'accéder à IR ou DR.

2.4) INTERFAÇAGE MODULE AFFICHEUR

Le module est prévu pour s'interfacer :
- directement sur un ou plusieurs ports d'un µC ou après une interface d'E/S sur un système à µP
- sur les bus de données, etc d'un système à µP ou d'un µC en mode étendu

2.4.1) DESCRIPTION DES SIGNAUX D'INTERFAÇAGE AVEC LE µP/µC

Les signaux d'interfaçage avec le µP/µC sont E, RS, R/W et DB0-DB7.

DB0-DB7 sont des E/S bidirectionnelles. Elles sont utilisées pour véhiculer les instructions et les données (codes des caractères à afficher, etc.). Leur comportement dépend des signaux RS et E.
Pour l'utilisation en mode 8 ou 4 bits, voir § 2.4.3)

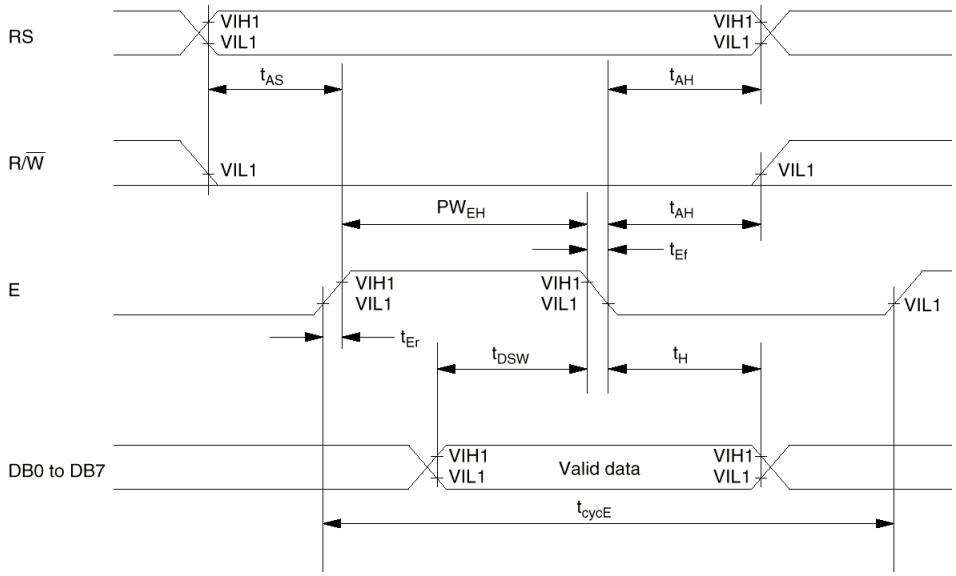
RS permet de choisir :

- la destination des données lors d'une écriture : registre instruction IR ou registre de données DR (*voir plus haut*).
- la provenance des données lors d'une lecture : données en provenance d'une des RAM ou adresse du compteur d'adressage des RAM

E est un signal de validation (Enable).
Lorsque E=0, les E/S sont en haute impédance.
Lorsque E=1, les E/S sont en entrée si on réalise une écriture (R/W=0) ou en sortie si on réalise une lecture (R/W=1).

E permet aussi de prendre en compte les signaux RS et R/W. Ceux-ci doivent donc être positionnés avant le front montant de E.
Lors d'une écriture les signaux en D0-D7 sont mémorisés sur le front descendant de E.

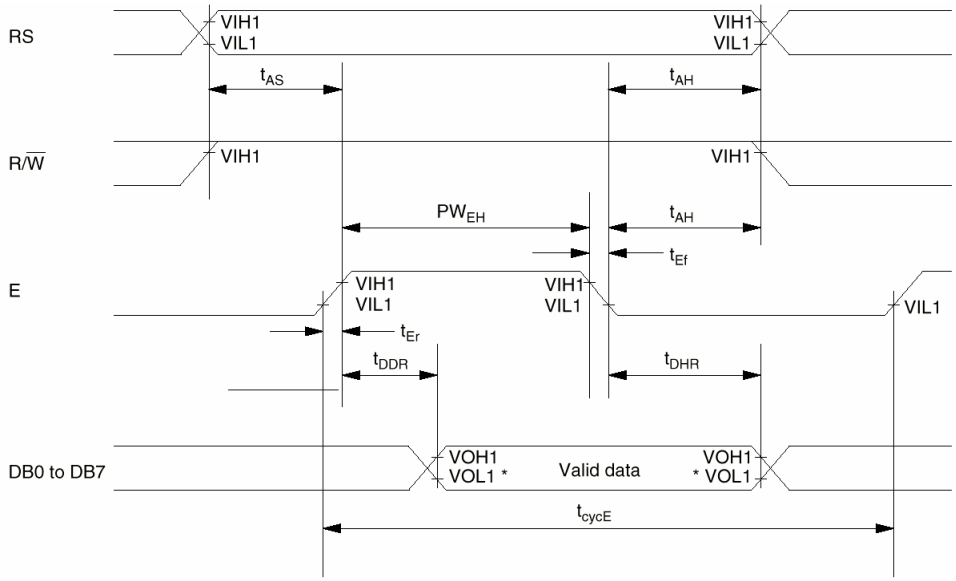
Les chronogrammes ci-dessous représentent les signaux lors d'une écriture.



« Valid data » correspond au code d'un caractère à afficher ou d'une instruction.

t_{AS} : durée de prépositionnement de RS et R/W par rapport au front montant de E.
t_{AH} : durée de maintien de RS et R/W par rapport au front descendant de E
t_{DSW} : durée de prépositionnement des données par rapport au front descendant de E
t_H : durée de maintien des données par rapport au front descendant de E

Les chronogrammes ci-dessous représentent les signaux lors d'une lecture :



« Valid data » correspond à une donnée renvoyée par le contrôleur du LCD (*voir plus loin, §2.6*)

t_{DDR} : retard à l'apparition des données par rapport au front montant de E
t_{DHR} : durée de maintien des données par rapport au front descendant de E

Les différentes durées selon les contrôleurs équivalents

Certaines durées changent avec la version du contrôleur. Par exemple la version HD44780U est plus rapide que la version HD44780S.

Les valeurs sont à vérifier

Réf des Contrôleurs / Paramètres		Samsung KS0066	Hitachi HD44780	Sanyo LC7985NA	Epson SED1278	OKI MSM6222	Durée recommandée[1]	Unité
Durée d'un cycle	t_{cycE}(min)	1000	1000	1000	500	667	1000	nS
Largeur E Niveau 1 Niveau 0	PW_{EH}(min) (min)	450 450	450 450	450 450	220 220	280 280	450 450	nS
Temps de montée E	t_{Er}(max)	25	25	25	25	25	25	nS
Temps de descente E	t_{Ef} (max)	25	25	25	25	25	25	nS
Prépositionnement signaux	t_{AS}(min)	40	140	140	40	140	140	nS
Prépositionnement données à l'écriture	t_{DSW}(min)	60	195	195	60	180	195	nS
Retard des données à la lecture	t_{DDR}(max)	320	320	320	120	220	320	nS
Maintien signaux	tA(max)	10	10	10	10	10	10	nS
Maintien donnée écriture lecture	t_H(min) t_{DHR}(min)	10 20	10 20	10 20	10 20	10 20	10 20	nS

1 : la durée recommandée permet d'utiliser n'importe quelle module afficheur LCD sans se soucier du contrôleur dont il est équipé.

2.4.2) INTERFAÇAGE DIRECT AVEC UN µP OU UN µC EN MODE ÉTENDU

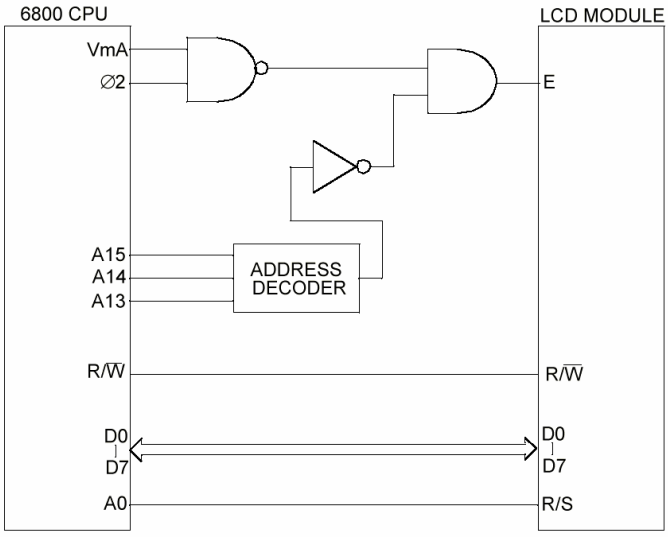
D0-D7 sont directement connectés sur le bus de données du µP.
Il faut reconstruire les signaux E, RS et R/W du contrôleur avec les signaux de contrôle du µP et un signal issu d'un décodeur d'adresse.

L'écriture d'une instruction dans le contrôleur du LCD s'effectue au niveau du µP en réalisant une écriture à une certaine adresse ; l'écriture du code d'une donnée à afficher s'effectue en réalisant une écriture à une autre adresse.
La lecture du registre d'état du contrôleur du LCD s'effectue au niveau du µP en réalisant une lecture à une adresse. La lecture du contenu du compteur d'adresse contrôleur de LCD s'effectue en réalisant une lecture à une autre adresse.

Les 2 adresses pour la lecture peuvent être ou non les même que celles de l'écriture. Ceci dépend des signaux de contrôle disponibles sur le µP employé.

Avec un µP rapide, il est nécessaire de rajouter des cycles d'attentes pour respecter les durées mentionnées ci-dessus.

Exemple d'interfaçage avec un µP 6800 ou un µC 68HC11 en mode étendu



Ici seules 2 adresses sont utilisées.
1 adresse permet d'accéder en lecture ou en écriture au registre instruction et une autre adresse permet d'accéder en lecture ou en écriture au registre de donnée.

VMA=1 uniquement lorsque les adresses sont valides.
ϕ2=1en fin d'un cycle machine, après positionnement de R/W.

Avec un µC du type 68HC11 en mode étendu, on dispose d'un signal E qui sert à construire le signal E du module LCD.
A compléter

Exemple d'interfaçage avec un 80C196 fonctionnant en mode étendu

L'interfaçage est plus compliqué que pour le cas précédent. Le µC ne possède pas de signal équivalent à ϕ2 du µC précédent. Il ne dispose pas non plus d'un signal unique R/W
Pour construire E du contrôleur, il est nécessaire d'utiliser un circuit combinatoire avec en entrée des bits d'adresses et les signaux /RD et /WR.
R/W du contrôleur est connecté à un bit d'adresse du µC, de même que RS.
On utilise ici 4 adresses distinctes pour commander l'afficheur.

Le détail de l'interfaçage ne peut être développé ici. Il faut ajouter des cycles d'attente, etc.

2.4.3) INTERFAÇAGE DIRECT AVEC UN µC OU L'INTERFACE PARALLÈLE D'UN SYSTÈME À µP

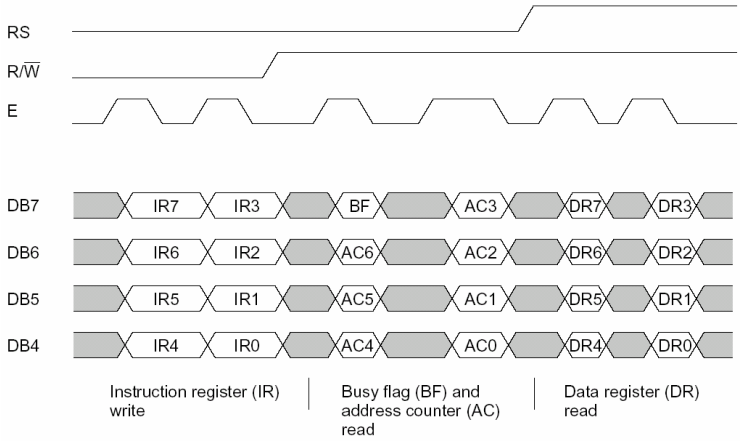
2 types d'interfaçages sont possibles : avec des données sur 8 ou 4 bits.

RS, E et R/W doivent être connectées sur des sorties du µC ; D0-D7 doivent être connectées sur des E/S ou sur des sorties si on ne souhaite pas effectuer de lecture.

Interfaçage avec un bus de données de 4 bits

Avec des données sur 4 bits, D0-D3 sont inutilisées. Il suffit de ne pas les connecter ; il est inutile de les fixer à un potentiel (des résistances de tirages doivent être placées dans le contrôleur).

Avec des données sur 4 bits, il faut d'abord transférer les 4 bits de poids forts puis les 4 bits de poids faible.
La figure suivante montre quelques échanges



On donne au § 2.8 la procédure d'initialisation avec un bus de données de 4 bits.

Structure type de la partie du programme pour un cycle d'écriture ou de lecture

Les temps mentionnés précédemment sont tels qu'avec un µC ordinaire (fréquence d'horloge < 1 dizaine de MHz, temps d'exécution d'une instruction > 200 à 300 ns) les programmes suivants conviennent.
On appelle P_E, P_RS et P_R/W les broches des ports sur lesquelles sont connectés les signaux E, RS et R/W. Le nom exact des broches du µC dépend de celui-ci.

Cycle d'écriture	Exemple en langage d'assemblage 68HC11
• Initialement P_E=0 • Positionner P_RS à la valeur voulue pour l'accès à IR ou DR et P_R/W à 0 • P_E = 1 • Placer les données sur les P_DBx *(cette instruction peut être placée n'importe où avant)* • P_E = 0	LDAA #VALEUR STAA PORTx ; PORTx utilisé pour E, R/W et RS BSET ....,...,... ; il faut d'abord charger un index LDAA #Donnees STAA PORTy; PORTy utilisé pour les données BCLR ...,...,....

Cycle de lecture	Exemple en langage d'assemblage 68HC11
• Initialement P_E=0 • Positionner P_RS à la valeur voulue pour l'accès à IR ou DR et P_R/W à 1 • Positionner P_E à 1 (1) • Lire les données sur les P_DBx • P_E = 0	LDAA #VALEUR STAA PORTx ; PORTx utilisé pour E, R/W et RS BSET ....,...,... ; il faut d'abord charger un index LDAA PORTy ; PORTy utilisé pour les données BCLR ...,...,....

1 : si utilisation d'un µC rapide, il faut attendre que les données soient disponibles avant d'effectuer la lecture.

Entre 2 cycles, il faut attendre un temps spécifié dans la documentation (dépend des versions du contrôleur).

2.5) AFFICHAGE AVEC LE MODULE DE 2 LIGNES DE 16 CARACTÈRES

Le module de 2 lignes de 16 caractères dispose d'une mémoire d'affichage (DDRAM) supérieure à la capacité d'affichage du panneau. Seule une partie des caractères correspondants aux codes en mémoire est affichée : c'est la fenêtre d'affichage.

2.5.1) FENÊTRE D'AFFICHAGE

Après une procédure de "RàZ" ou une instruction "retour en position initiale" (home), l'affichage part toujours des caractères situés chacun à la première adresse de chacune des 2 lignes de la DDRAM.

Lors de l'écriture dans la DDRAM, 2 possibilités selon l'instruction précédente :
- la fenêtre d'affichage est fixe (si le nombre de codes écrit excède la taille de la fenêtre, seule la première partie reste visible)
- la fenêtre d'affichage se déplace au fur et à mesure de l'écriture

Voir plus loin, entrée des codes des caractères

Déplacement de la fenêtre d'affichage.

La fenêtre peut se déplacer d'un caractère avec une instruction de décalage (shift). Il est possible de réaliser une animation avec plusieurs décalages et des temporisations.

La modification du compteur d'adresse de la DDRAM ne modifie pas la fenêtre d'affichage.

2.5.2) MÉMOIRE D'AFFICHAGE

Il est possible de remplir la mémoire avec les codes des caractères :
- uniquement pour la partie visible de l'afficheur
- en totalité. Dans ce cas, avec l'instruction de décalage, on peut réaliser une animation.

Si on n'utilise qu'une partie de la DDRAM, l'autre peut être utilisée comme de la RAM d'usage général, pour augmenter la capacité mémoire du µC.

Adresses de la mémoire DDRAM

Il y a 2 façons de considérer les adresses des mémoires :
- en ne considérant que l'adresse interne de chacune d'elle
- en considérant l'adresse externe totale délivrée par le µC (DB7 - DB0)

La documentation fait référence à la 1ère méthode. Si on veut accéder à la DDRAM, il faut forcer DB7 à 1 et son adresse interne est donnée par DB6-DB0. Si on veut accéder à la CGRAM, il faut forcer DB7, DB6 à 01 et l'adresse interne est donnée par DB5-DB0.

Si on considère l'adresse externe, celle-ci est formée par DB7-DB0.

Si l'affichage n'a pas suivi de décalage, les adresses internes / externes de la fenêtre d'affichage et de la partie invisible sont :

	fenêtre d'affichage (partie visible)	partie invisible
1ère ligne	adresses internes : 00h ... 0Fh (16 car) adresses externes : 80h... 8Fh	adresses internes : 10h ... 27h (24 car) adresses externes : 90h... A7h
2ème ligne	adresses internes : 40h ... 4Fh (16 car) adresses externes : C0h... CFh	adresses internes : 50h ... 67h (24 car) adresses externes : D0h... E7h

Il est possible de mémoriser 40 caractères pour chacune des lignes.

Pour le fonctionnement du décalage, voir §2.6) Les commandes du contrôleur d'affichage.

2.5.3) RAM DU GÉNÉRATEUR DE CARACTÈRE CGRAM

adresses externes : de 40h à 7fh. Adresses internes de 0 à 3F.

La CG-RAM, d'une capacité de 64 octets, offre 8 matrices de 5x8 points qui peuvent être programmées par l'utilisateur. Les codes à placer en DD-RAM pour l'accès aux caractères personnalisés vont de 00h à 07h. (Les codes images de 08h à 0Fh donnent le même résultat)

Programmation d'un caractère : chaque caractère occupe 8 octets. Pour chaque octet, seuls les 5 bits de poids faibles sont utiles, les autres étant à 0.
Un pixel devant apparaître en noir sera positionné à 1.

Le lien entre le code d'un caractère personnel et les adresses mémoire utilisées est donné dans le tableau suivant :

Code du caractère en DD-RAM	Adresses externes de la CG-RAM à programmer
00h	40h à 47h
01h	48h à 4Fh
02h	50h à 57h
03h	58h à 5Fh
04h	60h à 67h
05h	68h à 6Fh
06h	70h à 77h
07h	78h à 7Fh

Exemple : On désire créer le symbole flèche diagonale vers le haut avec comme code 00.

7	6	5	4	3	2	1	0	Code Hex
Non utilisé			■	■	■	■	■	$1F
			□	□	□	■	■	$03
			□	□	□	■	■	$03
			□	□	■	■	■	$07
			□	■	■	□	■	$0D
			■	■	□	□	■	$19
			■	□	□	□	□	$10

Pour programmer le caractère ci-contre il faut écrire dans la CG-Ram les codes : 1Fh a l'adresse 40h, 03h a l'adresse 41h, ..., et 10h a l'adresse 47h.

2.6) AFFICHAGE AVEC LE MODULE DE 4 LIGNES DE 20 CARACTÈRES

Le module de 4 lignes de 20 caractères dispose d'une mémoire d'affichage (DDRAM) égale à la capacité d'affichage du panneau.
Le module fonctionne comme un module 2 lignes dont toute la mémoire d'affichage est visible.
Les correspondances entre les lignes de l'afficheur et les adresses de la mémoire d'affichage sont données dans le tableau ci-dessous :

ligne	Adresses de la mémoire d'affichage
1	adresses internes : 00h ... 13h (20 caractères) adresses externes : 80h... 93h
2	adresses internes : 40h ... 53h (20 caractères) adresses externes : C0h... D3h
3	adresses internes : 14h ... 27h (20 caractères) adresses externes : 94h... A7h
4	adresses internes : 54h ... 67h (20 caractères) adresses externes : D4h... E7h

Les adresses de la ligne 3 suivent les adresses de la ligne 1 et les adresses de la ligne 4 suivent celles de la ligne 2.
Si le programme chargé d'écrire dans le LCD est mal conçu et qu'il écrit trop de caractères sur la ligne 1, l'écriture se poursuit ligne 3.

La notion de fenêtre d'affichage n'a pas beaucoup de sens ici. Il est cependant possible de réaliser des décalages, à condition de bien réfléchir au résultat (caractères changeant de lignes après un décalage, etc).

Tout le § 2.5.3) s'applique ici.

2.7) LES COMMANDES DU CONTRÔLEUR D'AFFICHAGE

Les instructions et les données ne peuvent être écrites ou lues que si les précédentes ont été correctement traitées. 2 façons de procéder :

- scruter le bit d'état en réalisant une lecture de IR et n'autoriser une écriture que si ce bit est à 0 (module prêt).
- attendre avec une temporisation le temps spécifié dans la documentation constructeur.

L'écriture des instructions se fait avec RS=0 (voir plus haut). Les différentes instructions sont différentiées par les valeurs de DB7-DB0. R/W est toujours égal à 0 puisqu'il s'agit uniquement d'écriture.
L'écriture et la lecture des données s'effectuent avec RS=1.

Le tableau ci-dessous résume les commandes du contrôleur d'affichage.

Commande	RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0	Description	Durée d'exécution[1]
Effacement de l'affichage	0	0	0	0	0	0	0	0	0	1	Efface l'ensemble de la mémoire d'affichage sans toucher au générateur de caractère. Met le curseur en position Home, à l'adresse 00.	1,64ms
Curseur en position « Home »	0	0	0	0	0	0	0	0	1	*	Met le curseur en position Home. Si l'affichage à été décalé, il est remis à sa position d'origine.	1,64ms
Mode d'entrée	0	0	0	0	0	0	0	1	ID	S	Détermine le changement d'adresse après écriture d'un caractère **(ID)** et le déplacement collectif d'une position de l'ensemble de l'affichage **(S)**.	40µs
	ID = 1 : Incrémentation / ID = 0 : Décrémentation S = 1 : Décalage de tout l'affichage											
Mode d'affichage	0	0	0	0	0	0	1	D	C	B	Met l'affichage en ou hors fonction **(D)**. Met le curseur en ou hors fonction **(C)**. Fait clignoter le caractère situé au-dessus du curseur **(B)**, clignotement se traduisant par une alternance du caractère et du caractère FF (rectangle noir)	40µs
	D = 1 : affichage en fonction C = 1 : curseur visible B = 1 : clignotement du caractère											
Décalage affichage / Déplacement curseur	0	0	0	0	0	1	S/C	R/L	*	*	Déplace le curseur ou l'ensemble de l'affichage sans modifier le contenu de la mémoire.	40µs
	S/C=1 : décalage de l'affichage S/C=0 : déplacement du curseur R/L=1 : décalage à droite R/L=0 : décalage à gauche											
Mode de fonctionnement	0	0	0	0	1	DL	N	F	*	*	Fixe la largeur du bus de données (**DL**), le nb de lignes de l'afficheur (**N**) et la taille de la matrice des caractères (**F**)	40µs
	DL=1 : 8 bits / DL=0 : 4 bis N=1 : 2 lignes / N=0 : 1 ligne F=1 : 5*10 pixels / F=0 : 5*7 pixels											
Écriture de l'adresse de la CGRAM	0	0	0	1	Adresse interne de la CGRAM (de 0 à 3F)						Définit l'adresse de la mémoire du générateur de caractères. Les données peuvent être lues ou écrites à cette adresse	40µs
Écriture de l'adresse de la DDRAM	0	0	1	Adresse interne de la DDRAM (de 0 à 7F)							Définit l'adresse de la mémoire de données. Les données, correspondant aux codes des caractères ASCII, peuvent être lues ou écrites à partir de cette adresse.	40µs
Lecture de l'indicateur Busy et de l'adresse	0	1	BF	Valeur courante du compteur d'adressage							Lit l'indicateur Busy **(BF)** pour vérifier que l'afficheur et en mesure de traiter la commande suivante et l'adresse courante du compteur d'adressage.	1µs
	BF=1 : Occupé / BF=0 : le contrôleur peut accepter une nouvelle commande ou donnée											
Écriture des données	1	0	Donnée à écrire								Écriture dans la CGRAM ou DDRAM selon la valeur du compteur d'adressage	46µs
Lecture des données	1	1	Donnée lue								Lecture dans la CGRAM ou DDRAM selon la valeur du compteur d'adressage	46µs

1 : les durées peuvent être plus importantes avec certains contrôleurs anciens (jusqu'à 3 fois les valeurs mentionnées ici).

On ne peut écrire les données dans une des 2 RAM qu'après avoir :

- choisi le bon mode d'entrée
- donné l'adresse de début de l'enregistrement des données (positionnement du compteur d'adressage)

Comportement de l'affichage lors d'une écriture des données.

Lors de l'écriture des données (codes des caractères) dans la DDRAM, le comportement de l'affichage dépend de la programmation précédente.

Exemple.
affichage correctement initialisé
mode d'entrée : Incrémentation du compteur d'adressage (décalage à gauche du curseur), pas de décalage de l'affichage
affichage et curseur "on"
effacement de l'affichage (compteur d'adressage à 0)

Lors de chaque écriture de donnée en DDRAM, le curseur est décalé et le nouveau caractère est affiché à la suite des autres.

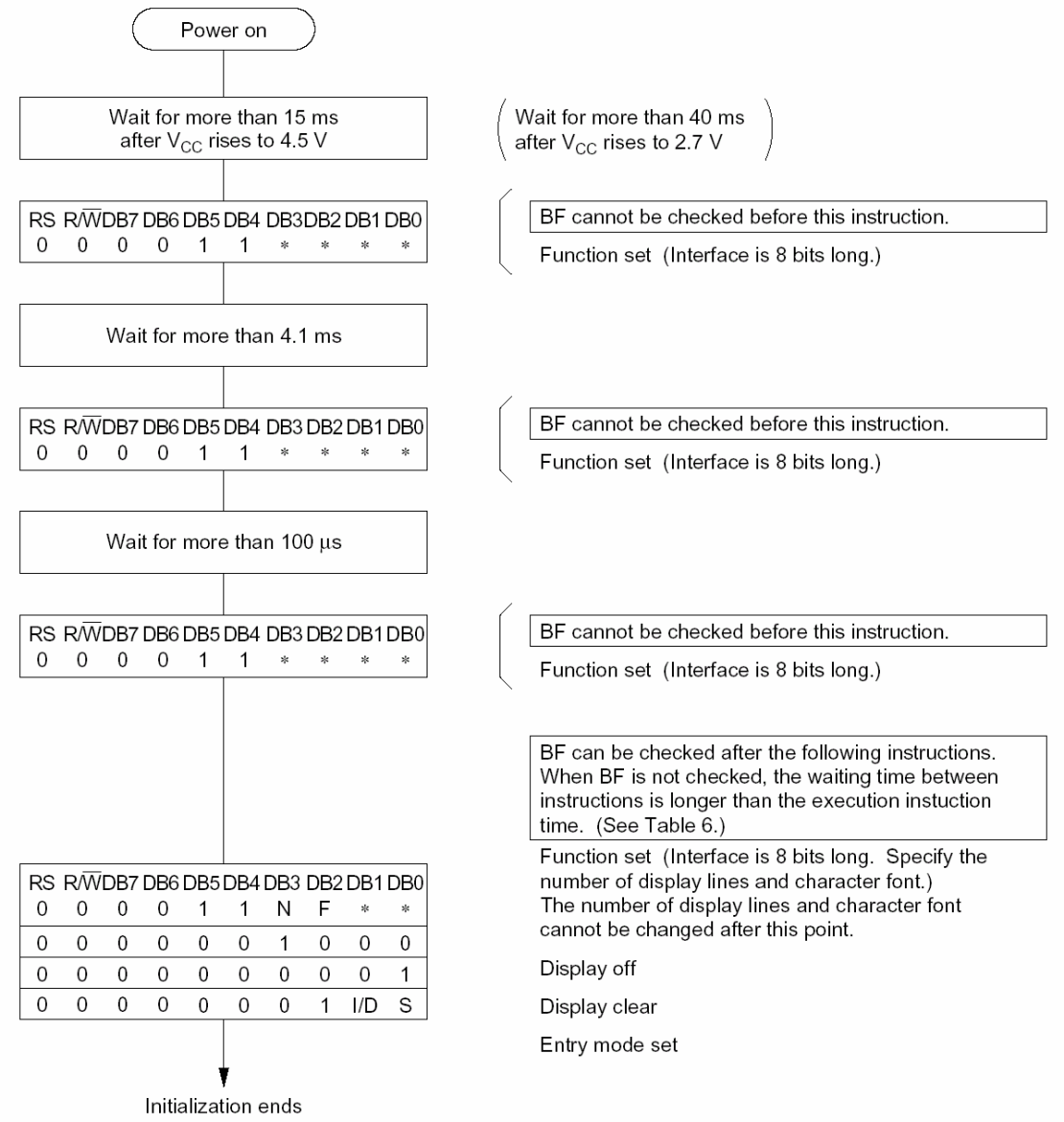
Fonctionnement du décalage

A chaque commande de décalage, l'affichage est décalé. Après le caractère du dernier emplacement mémoire DDRAM (le 40ème), c'est le caractère du premier emplacement mémoire DDRAM qui est affiché à la suite. Ceci permet de réaliser des animations circulaires.

2.8) INITIALISATION DU MODULE

BUS DE DONNÉES DE 8 BITS

Pour un module de 2 lignes, fonctionnant en mode 8 bits, le constructeur préconise les opérations suivantes (les temps dépendent des versions du contrôleur) :

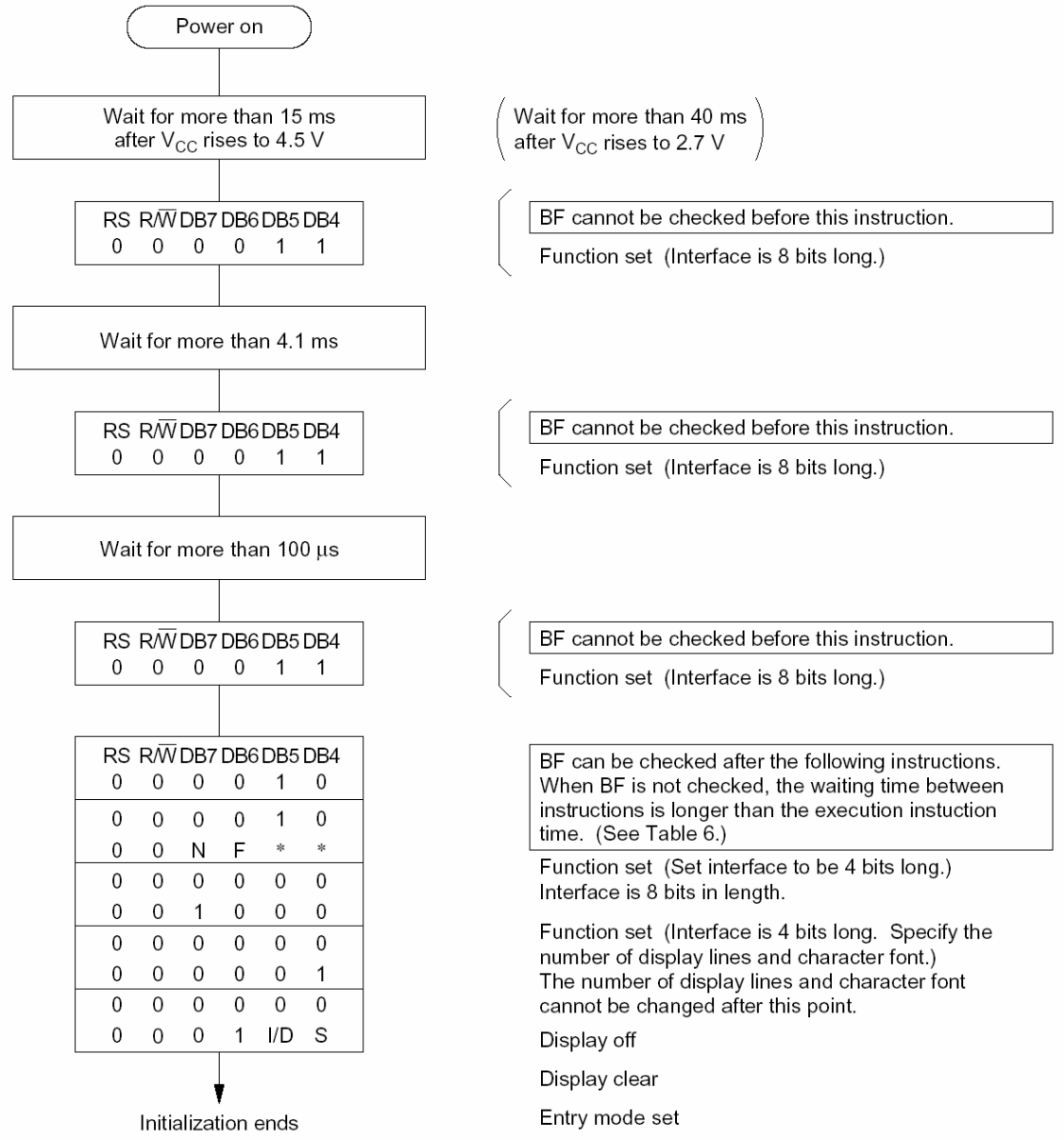


- attendre au minimum15 ms après l'établissement de la tension d'alimentation à 4,5V min
- écrire 3xh dans IR
- attendre 4,1 ms min
- écrire 3xh dans IR
- attendre 100 µs min
- écrire 3xh dans IR
- attendre 40 µs min
- écrire 38h dans IR (2 lignes, matrice 5x8)
- attendre la fin de l'opération (40µs)
- écrire 01h dans IR (effacement de l'écran)
- attendre la fin de l'opération (qqs ms, doc peu précise)
- écrire 0Ch dans IR (Affichage en fonction, curseur non visible, pas de clignotement du caractère)
- attendre la fin de l'opération (40µs)
- écrire 06h dans IR (Incrémentation du compteur d'adressage après écriture d'un caractère)
- attendre la fin de l'opération (40µs)

Pour attendre la fin de l'opération, on peut tester le bit BF

BUS DE DONNÉES DE 4 BITS

Pour un module de 2 lignes, fonctionnant en mode 4 bits, le constructeur préconise les opérations suivantes (les temps dépendent des versions du contrôleur) :



- attendre au minimum 15 ms après l'établissement de la tension d'alimentation à 4,5V min
- écrire 3h dans IR
- attendre 4,1 ms min
- écrire 3h dans IR
- attendre 100 µs min
- écrire 3h dans IR
- attendre 40 µs min
- écrire 2h dans IR (interface en mode 4 bits)
- attendre la fin de l'opération (40 µs)
- écrire en 2 fois 28h dans IR (2 lignes, matrice de 8x5)
- attendre la fin de l'opération (40 µs)

- écrire 01h dans IR (effacement de l'écran)
- attendre la fin de l'opération (qqs ms, doc peu précise)
- écrire en 2 fois 0Ch dans IR (Affichage en fonction, curseur non visible, pas de clignotement du caractère)
- attendre la fin de l'opération (40µs)
- écrire en 2 fois 06h dans IR (Incrémentation du compteur d'adressage après écriture d'un caractère)
- attendre la fin de l'opération (40µs)

Pour attendre la fin de l'opération, on peut tester le bit BF

2.8) SOUS PROGRAMMES (FONCTIONS EN LANGAGE C) NÉCESSAIRES POUR LA GESTION COMPLÈTE DE L'AFFICHEUR

PRÉSENTATION GÉNÉRALE

Des fonctions de haut niveau permettent d'afficher une ligne entière à partir de la gauche de l'afficheur ou d'afficher une chaîne de caractères à une position déterminée.
Ces fonctions de haut niveau utilisent des fonctions de bas niveau pour fixer la valeur du compteur d'adressage, écrire une commande, etc.

Pour respecter les durées entre écritures de commande, etc., il est possible d'utiliser des temporisations entre les commandes ou des attentes avec test de l'état du contrôleur.
Il est possible d'incorporer ou non ces temporisations ou attentes dans les fonctions de bas niveau ou de niveau intermédiaire.

Le tableau ci-dessous donne un exemple d'ensemble de fonctions permettant de gérer totalement l'affichage.

Fonction	Catégorie	Description
void **EcritUneLigne**(const unsigned char* PtrCarMsg, unsigned char Ligne) PtrCarMsg : Pointeur sur un caractère du message, initialement sur le 1er Ligne : N° de la ligne d'affichage	haut niveau	Permet d'écrire une ligne complète en écrasant le contenu précédent. Si le message fait moins de 16 ou 20 caractères, des blancs sont insérés en fin. Avec un module de 2 lignes, le message peut faire plus de 16 caractères. La fonction de décalage doit être utilisée pour un affichage complet.
void **EffaceUneLigne**(unsigned char Ligne)	haut niveau	Permet d'effacer une ligne avant d'écrire un ou plusieurs caractères ou une chaîne. En fin de fonction, le compteur d'adressage est remis en début de ligne.
void **EcritChaine**(const unsigned char* PtrCarMsg)	haut niveau	Ecrit une chaîne à partir de la position courante du compteur d'adressage
void **PosCmptMemAff**(unsigned char Position, unsigned char Ligne) Position : position par rapport à la gauche de l'afficheur. 1er caractère = 0	bas niveau	Fixe la valeur du compteur d'adressage en fonction de la position et de la ligne indiquées

void **EcritCar** (unsigned char Code)	bas niveau	Écrit un caractère à partir de la position courante du compteur d'adressage
void **EcritCde**(unsigned char Valeur)	bas niveau	Écrit une commande (défilement, etc.)
void **AttenteFinOperation**(void)	bas niveau	Attend que le drapeau BF passe à 1, indiquant ainsi que l'opération en cours est terminée.

E, RS ET RW

Les signaux E, RS, RW et BF utilisés ci-dessous peuvent être :
- définis avec des directives #define
- ou remplacés par des expressions réalisant un masquage
- ou correspondre à des champs de bits

Utilisation de directive #define

Selon le compilateur et le µC utilisé, on peut utiliser les définitions suivantes :

Compilateur disposant du type bit (extension C ANSI)
ex : compilateur HiTech pour µC PIC de Microchip
#define E RC2 RC2 est déclaré dans le fichier en-tête comme une variable de type bit

Compilateur ne disposant pas du type bit
Il faut utiliser un champ de bit
ex : #define E RC.b2 Avant utilisation, RC doit être déclaré comme un champ de bit

Utilisation de masquage

Pour E = 1, il faut utiliser E=PORTx | MASQUE_E_1 ;
MASQUE_E_1 comporte un seul bit à 1 : celui qui correspond à la position de E dans le PORTx
Pour E = 0, il faut utiliser E = PORTx & MASQUE_E_0 ;
MASQUE_E_0 comporte un seul bit à 0 : celui qui correspond à la position de E dans le PORTx

DÉTAIL DES FONCTIONS

Étant donné que RS et RW sont dans la plupart des commandes à 0, on prend cette valeur comme la valeur par défaut. Lorsque une commande nécessite un changement d'état, ce changement s'effectue dans la fonction à utiliser. Cette fonction remet en fin RS ou RW à 0.

Dans ce qui suit, on suppose que les données sont connectées sur le PortDonnee. Le sens de transfert sur ce port est fixé avec un registre de direction SensPortDonnee. 1 correspond à la mise en entrée
Le compilateur utilisé dispose d'une extension au C ANSI pour la notation binaire. RS, RW et E sont définies chacune comme une broche du µC. Avec un compilateur ne disposant pas de type booléen, il faut procéder par masquage.

```
void AttenteFinOperation(void)
{
unsigned char VarTemp;
SensPortDonnee = 0b11111111; // les broches de données du module sont toutes en sorties
                             // lors d'un opération de lecture, donc les broches du µC sont en entrée
RW=1;
do
        {
```

```
        E = 1;
        asm("nop"); // pour utilisation avec µC rapide
        asm("nop");
        VarTemp=PortDonnee & 0b10000000;
        E=0 ;
        }
        while(VarTemp !=0);
RW=0;
SensPortDonnee=0 pour revenir à la configuration initiale
}
```

```
void EcritCde(unsigned char Valeur)
{PortDonnee = Valeur;
E = 1;
asm("nop"); // pour assurer une durée suffisante à l'état 1 de E avec un µC rapide
E = 0;
Tempo(D100us); // peut être remplacé par AttenteFinOperation()
}
```

```
void EcritCar (unsigned char Code)
{PortDonnee = Code;
RS = 1;
E = 1;
asm("nop"); // pour permettre de respecter la durée à l'état haut de E avec un µC rapide
E = 0;
RS = 0 ;
Tempo(D100us); // peut être remplacé par AttenteFinOperation()
}
```

```
void PosCmptMemAff(unsigned char Position, unsigned char Ligne)
                // pour un module de 2 lignes de 16 caractères
{if (Ligne==1)
        EcritCde(0x80 + Position); /*0x80 adresse de la DDRAM correspondant au caractère le
                                    plus à gauche */
if (Ligne==2)
        EcritCde(0xC0 + Position);
}
```

```
void EcritUneLigne(const unsigned char* PtrCarMsg, unsigned char Ligne)
        /* const pour accès à données en ROM et RAM, syntaxe propre à un compilateur pour
        un µC PIC */

{
unsigned char NbCarTransfere=0;

PosCmptMemAff(0, Ligne);
while ((*PtrCarMsg != 0) && (NbCarTransfere != CARS_MAX))
        /* CARS_MAX est égal au nombre de caractères max de la mémoire d'affichage d'une li-
        gne moins un. Ex : CARS_MAX = 39 */
```

```
        {
        EcritCar(*PtrCarMsg );
        NbCarTransfere ++;
        PtrCarMsg ++;
        }
while (NbCarTransfere != CARS_MAX)
        {
        EcritCar(0x20); /*caractere espace*/
        NbCarTransfere ++;
        }
}
```

Autre variante :

```
void EcritUneLigne(const unsigned char* PtrCarMsg, unsigned char Ligne)
        /* const pour accès à données en ROM et RAM, syntaxe propre à un compilateur pour
        un µC PIC */
{
unsigned char NbCarTransfere=0;
PosCmptMemAff(0, Ligne);
if (*PtrCarMsg!=0)
        {do
                {
                EcritCar(*PtrCarMsg);
                NbCarTransfere ++;
                PtrCarMsg ++;}

        while ((*PtrCarMsg != 0) && (NbCarTransfere != CARS_MAX));
        /* CARS_MAX est égal au nombre de caractères max de la mémoire d'affichage d'une li-
        gne. Ex : CARS_MAX = 40 */
        }
while (NbCarTransfere != CARS_MAX)
        {EcritCar(0x20); /*caractere espace*/
        NbCarTransfere ++;
        }
}
```

```
void EffaceUneLigne(unsigned char Ligne)
{
unsigned char NbCarTransfere=0;
PosCmptMemAff(0, Ligne);

while (NbCarTransfere != CARS_MAX)
        {EcritCar(0x20); /*caractere espace*/
        NbCarTransfere ++;
        }
PosCmptMemAff(0, Ligne);
}
```

```
void EcritChaine(const unsigned char* PtrCarMsg)
{
while (*PtrCarMsg != 0)
        {EcritCar(*PtrCarMsg);
```

```
        PtrCarMsg ++;
        }
}
```

L'implémentation de **Tempo** est étroitement liée au µC utilisé. L'en-tête est :

```
void Tempo(unsigned char Val)
```

FONCTIONS POUR LA CONVERSION D'UN NOMBRE EN UNE SUITE DE CODES ASCII

```
unsigned char* CodageIntASCII(int Valeur)
{
static unsigned char CodesASCII[5];
unsigned char Position = 0,
                Quotient;
unsigned int Diviseur = 10000;
unsigned char ChiffreSignificatif = FAUX;

if (Valeur < 0) {
        Valeur = - Valeur;
        CodesASCII[Position++]='-';
        }

if (Valeur == 0){
        CodesASCII[Position++] = '0';
}
else {
        while (Diviseur != 0) {
                Quotient = Valeur / Diviseur;
                if ((ChiffreSignificatif == VRAI)||(Quotient!=0)) {
                        CodesASCII[Position++]='0'+Quotient;
                        ChiffreSignificatif = VRAI;
                }
                Valeur = Valeur % Diviseur; // reste de la division
                Diviseur = Diviseur /10;
        }
}
CodesASCII[Position] = NULL;
return(CodesASCII);

}
```

Pour une conversion d'un entier non signé vers une chaîne de caractère, il suffit de conserver qu'une partie de ce programme.

2.9) CARACTÈRES EN MÉMOIRE ROM

Il existe plusieurs versions de contrôleur. La plus répandue est celle avec les caractères suivants en mémoire.

3) AFFICHEUR À CRISTAUX LIQUIDES : CONSTITUTION ET COMMANDES

Ce qui suit est utile pour la mise en œuvre des afficheurs LCD et de leurs circuits de commande.

3.1) CONSTITUTION ÉLECTRIQUE D'UN AFFICHEUR

Chaque élément de l'afficheur (segment, pictogramme, point, etc.) nécessite 2 surfaces conductrices (électrodes) en regard.
Pour le détail sur le principe de fonctionnement d'un afficheur à cristaux liquides, voir l'annexe.

Une des surfaces est située sur la face arrière de l'afficheur : « backplane »

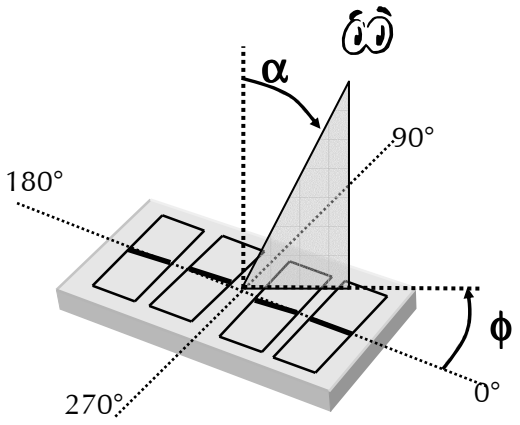
Selon le nombre d'éléments à afficher, on rencontre 2 types de configurations :
- 1 commun (backplane) pour tous les éléments + 1 électrode pour chaque élément
- plusieurs communs + 1 électrode pour chaque groupe d'éléments interconnectés

La dernière méthode permet de réduire le nombre de broches de l'afficheur.
Les commandes dépendent directement de l'organisation des électrodes.

3.2) CONTRASTE ET TEMPS DE RÉPONSE

Le contraste dépend de :
- la position relative afficheur / observateur
- la valeur efficace de la tension appliquée entre le commun et une électrode (voir § suivant)

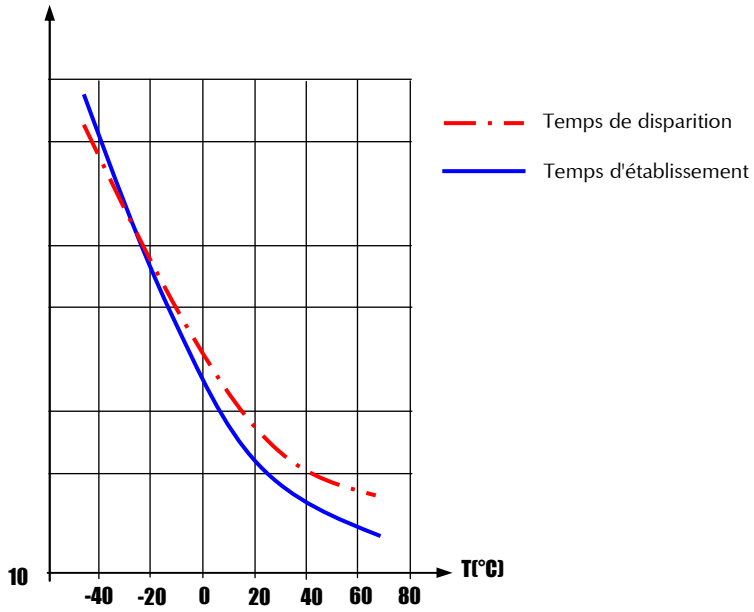
La position relative afficheur / observateur est décrite avec l'angle de visée α et la direction de visée ϕ. α est considéré comme "angle Zénith" des perpendiculaires à la surface et ϕ comme azimut de la direction "3 heures".



Le contraste diminue quand l'angle α augmente.

Après la mise sous tension, un temps de retard à la mise en fonctionnement s'écoule jusqu'à ce qu'une variation de contraste apparaisse. Le temps de montée est mesuré entre 10% et 90% du contraste. De la même manière, il existe un temps de retard ou de retombée.
Les temps de réponse sont fonction de la viscosité de la substance ainsi que de l'épaisseur de la cellule : la viscosité et les temps de réponse décroissent quand la température croit. A 25°C, les

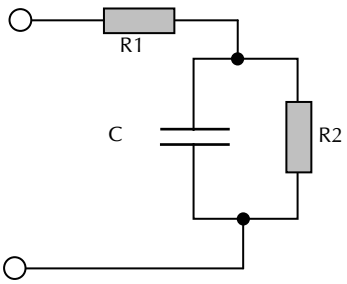
temps de réponse sont de l'ordre de 40 ms pour le temps d'établissement et de l'ordre de 40 à 80ms pour le temps de disparition. Ce paramètre représente un des gros inconvénients des LCD.



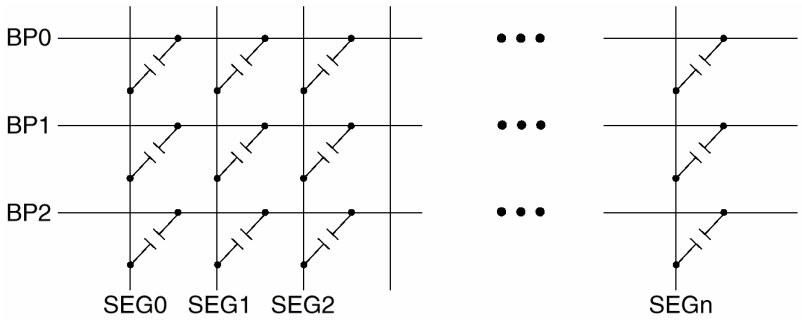
3.3) COMMANDE D'UN AFFICHEUR LCD

MODÈLE ÉQUIVALENT

Le modèle équivalent entre 2 électrodes d'un élément est le suivant :



Pour un afficheur constitué de segments (SEG) à plusieurs communs (BP), si on néglige les résistances, on obtient le modèle équivalent suivant :



TENSION DE COMMANDE

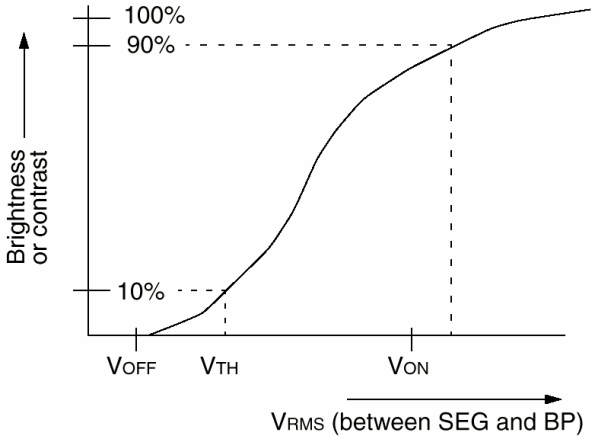
Un afficheur transmet la lumière lorsqu'aucune différence de potentiel n'est appliquée entre les 2 électrodes d'un élément. L'élément à afficher n'est pas visible.
Il ne transmet pas la lumière lorsqu'une différence de potentiel est appliqué entre les 2 électrodes. L'élément à afficher est visible en noir.

La tension entre électrodes doit toujours être à valeur moyenne proche de 0 (inférieure à 50mV) pour éviter de détruire les cristaux liquides par électrolyse, dans le cas où les électrodes ne sont pas isolées des cristaux liquides.
Pour disposer d'une tension à valeur moyenne nulle, on envoie des signaux rectangulaires ou en marches plus ou moins complexes entre les électrodes. Ces signaux sont détaillés dans les 2 § suivants.

C'est la valeur efficace de la tension entre les électrodes qui détermine le comportement de l'afficheur.

Pour une bonne visibilité, il faut :

- **Voff (eff) < Vth**
- **Von (eff) >> Vth**
- **D=Von (eff) / Voff (eff) le plus grand possible. D est le rapport de contraste (Discrimination ratio)**



Un Von trop grand réduit cependant la durée de vie de l'afficheur.
Les différentes valeurs Voff, Vth, Von varient d'un afficheur à l'autre.

La fréquence des signaux appliqués entre les électrodes est limitée par 2 caractéristiques :

- le scintillement de l'affichage si la fréquence est trop basse
- une consommation excessive si la fréquence est trop haute

Le phénomène de scintillement apparaît pour des fréquences inférieures à 30 Hz. La fréquence supérieure est généralement de l'ordre de 200 Hz.
L'application de tensions continues supérieures à 50 mV n'est pas admissible, parce que ces tensions provoqueraient alors des réactions d'électrolyse sur le cristal liquide qui peuvent réduire fortement la durée de vie (segment qui reste visible en permanence).

COMMANDE D'UN AFFICHEUR AVEC UN SEUL COMMUN (BACKPLANE)

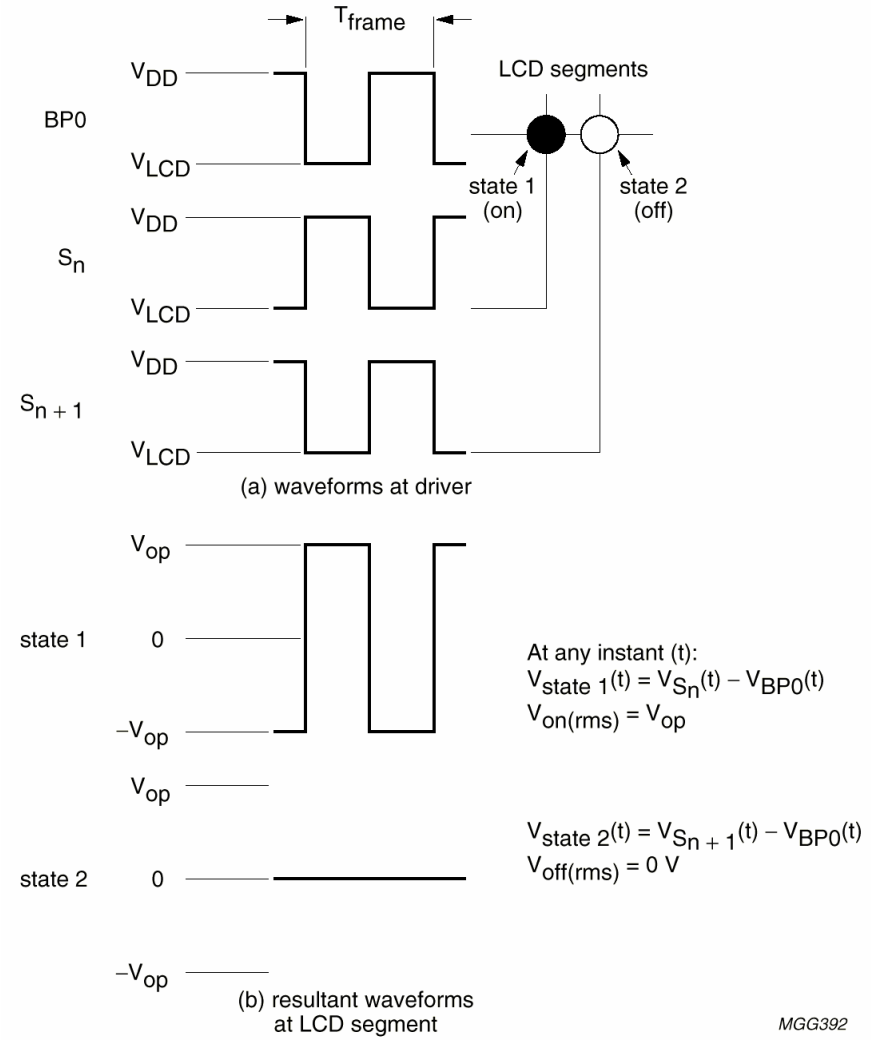
Quand il n'y a qu'un seul commun, la commande est dite **statique**.

La figure suivante montre les signaux appliqués sur les électrodes et les tensions résultantes entre les électrodes correspondant à un

Voff = 0
Von = VDD-VLCD

VLCD permet de faire varier Von.

D est infini. Le contraste est très bon



COMMANDE D'UN AFFICHEUR AVEC PLUSIEURS COMMUNS

Avec plusieurs communs, l'affichage est **multiplexé**. Les éléments sont commandés par groupes.

Pour respecter les contraintes présentées auparavant, plusieurs formes de tensions peuvent être appliquées sur les électrodes. Elles présentent toutes des marches d'escalier.

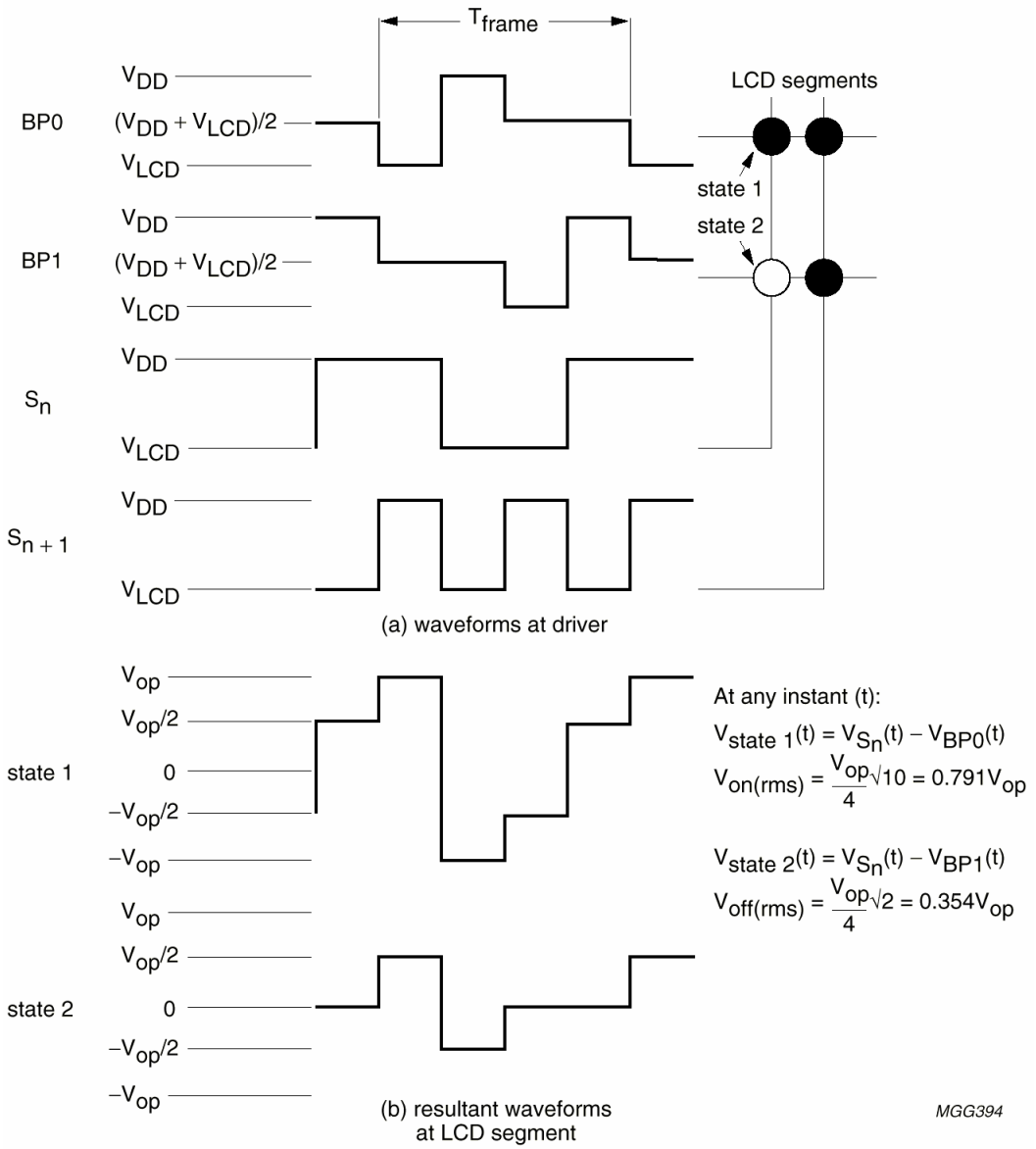
Les différentes possibilités se distinguent par :

- le nombre de valeurs possibles des tensions : 3 ou 4 (2 ou 3 marches) (LCD bias configuration)
- le taux de multiplexage qui dépend du nombre de communs (BP = Back Plane) (LCD drive mode)

On présente ci-dessous uniquement 2 cas :

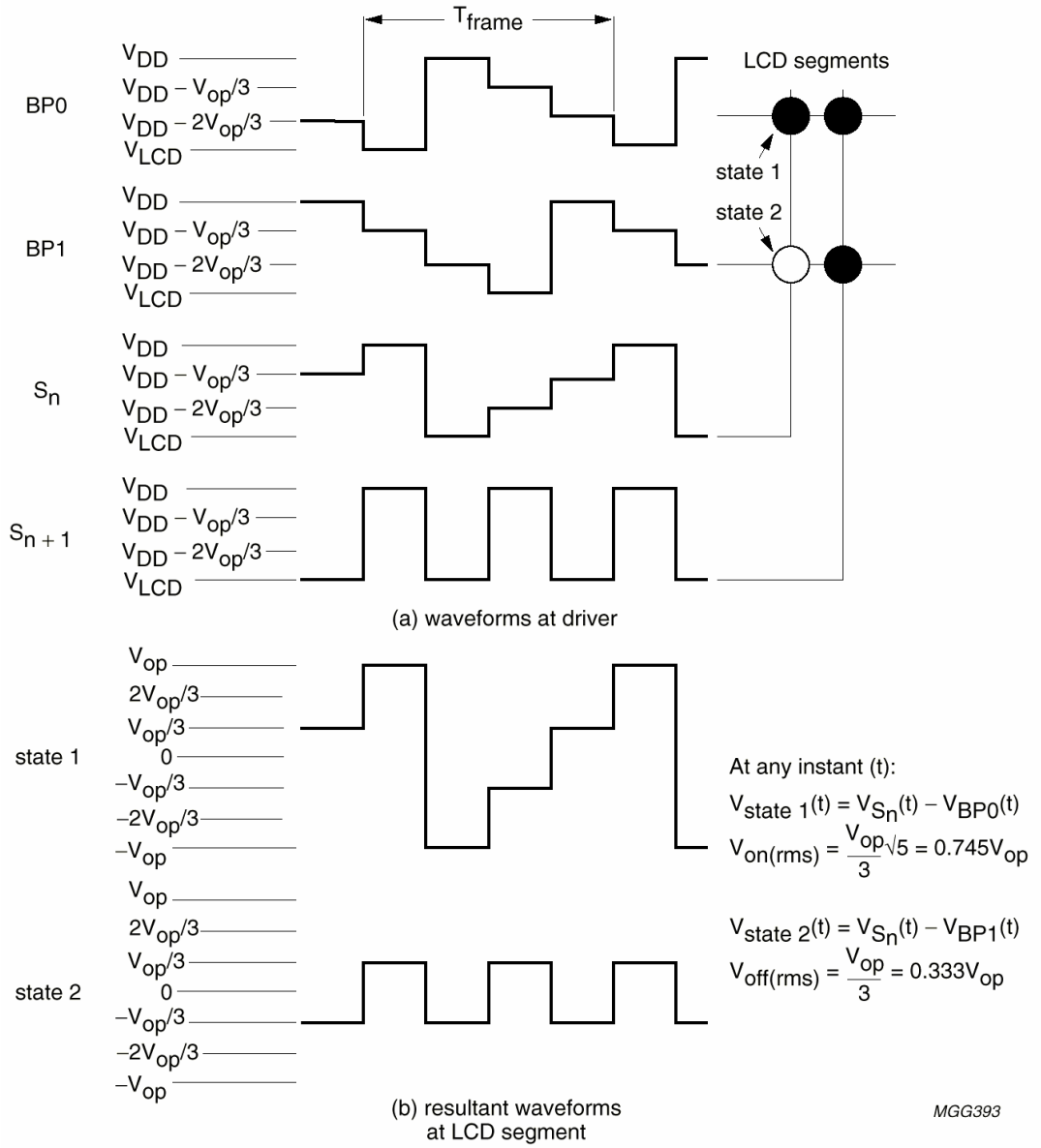
- multiplexage 1 : 2 (2 communs) / 3 niveaux
- multiplexage 1 : 2 (2 communs) / 4 niveaux

multiplexage 1 : 2 / 3 niveaux (1/2 bias)



Vop = VDD – VLCD D = 2,236

multiplexage 1 : 2 / 4 niveaux de tension (1/3 bias)



D = 2,236 Vop = VDD - VLCD

Généralisation

VLCD est obtenu à partir d'un montage avec un potentiomètre. Les tensions de polarisation sont obtenues avec un diviseur de tension. VLCD permet de faire varier Von et Voff et donc le contraste.
Dans les figures ci-dessus Vop = VDD – VLCD.

Plus le nombre de communs est grand (taux de multiplexage grand), plus le rapport D est faible.
VLCD doit être finement ajusté pour obtenir le meilleur contraste possible.
Un afficheur de 2 lignes de 16 caractères fonctionne avec un multiplexage 1 : 12.

4) µC ET CI SPÉCIALISÉS POUR L'INTERFAÇAGE DE LCD

4.1) GÉNÉRALITÉS

Les µCs et CIs spécialisés délivrent les signaux présentés au §3.2)

Ils se distinguent notamment par leur possibilité de configuration ou non.
Les CIs configurables permettent de commander en général un grand nombre d'éléments sur un LCD. Ils ont donc un nombre important de broches.

Exemples de circuits configurables :

Le CI PCF8566 de Philips peut commander un LCD de 1 à 4 communs avec 24 (1 commun) à 96 éléments (4 communs). Avec un LCD à 1 commun, la commande est statique. Lorsque le LCD à 2 communs, il est possible de choisir entre 3 et 4 niveaux de commande. Le taux de multiplexage est fonction du nombre de communs.
Le µC spécialisé PIC16C92x peut commander un LCD de 1 à 4 communs avec 32 (1 commun) à 116 éléments (4 communs). Avec un LCD à un commun, la commande est statique. Lorsque plusieurs communs sont utilisés, il y a 4 niveaux de commandes.

Exemples de circuit non configurable

Le CI MSM6544 de Oki peut commander un LCD de 2 communs max et 80 éléments max. La commande est toujours multiplexée.
Le CI ICM7211 d'Intersil est prévu pour commander un LCD de 4 chiffres 7 segments avec un seul commun
Le circuit de la famille CMOS 4000 4543 permet de commander un LCD d'un seul chiffre 7 segments. *Ce circuit est obsolète.*

Les sorties des µCs et CIs spécialisés se connectent directement au LCD.
Les circuits à commande uniquement statique ne nécessitent pas de tension variable VLCD.

Certains CI ont besoin de composants externe pour fabriquer une horloge nécessaire au bon fonctionnement.

L'élaboration des signaux de commande du LCD est transparente à l'utilisateur. Celui-ci doit uniquement :
- écrire dans des registres internes les états des éléments du LCD
- éventuellement programmer un fois la configuration souhaitée (nb de communs, etc.)

4.2) INTERNCONNEXION CI SPÉCIALISÉ - µC / ÉCRITURE DES ÉTATS DES ÉLÉMENTS

Les différentes informations à envoyer du µC au CI spécialisé peuvent être sous forme série ou parallèle.
Les CIs configurables ont en général un grand nombre de broches pour la commande du LCD. Pour diminuer le nombre de broches total, les informations sont en général envoyées en série.
Certains CIs non configurables ont un nombre limité de broches vers le LCD (<30). On peut dans ce cas utiliser une liaison parallèle.

Pour l'affichage, on peut envoyer :

- les codes des caractères, chiffres ou lettres, à afficher dans le cas CI prévu pour commander uniquement un affichage alphanumérique
- les états de tous les éléments du LCD dans le cas d'un CI pour commander un affichage quelconque (caractères alphanumérique, dessins, etc.)

On prend ici 2 exemples :

- un CI prévu pour commander 4 chiffres de 7 segments : ICM7211
- un CI configurable PCF8566

Seules quelques grandes lignes sont mentionnées ici. Pour plus de détail, voir la documentation complète de ces circuits.

CI POUR COMMANDE DE 4 CHIFFRES : ICM7211

Les informations à mémoriser sont les 4 codes DCB des chiffres à afficher. Des décodeurs internes DCB → 7 segments élaborent les informations utilisées pour la commande des segments.
Les 4 codes DCB sont envoyés en // les uns après les autres. Des signaux permettent de sélectionner le chiffre dont le code DCB est transmis.

CI CONFIGURABLE PCF8566

Les informations de configuration et les états des éléments du LCD sont transmis sous forme série. Avec le circuit présenté, le protocole de communication est l'I2C. Il existe des circuits du même type avec d'autres protocoles.

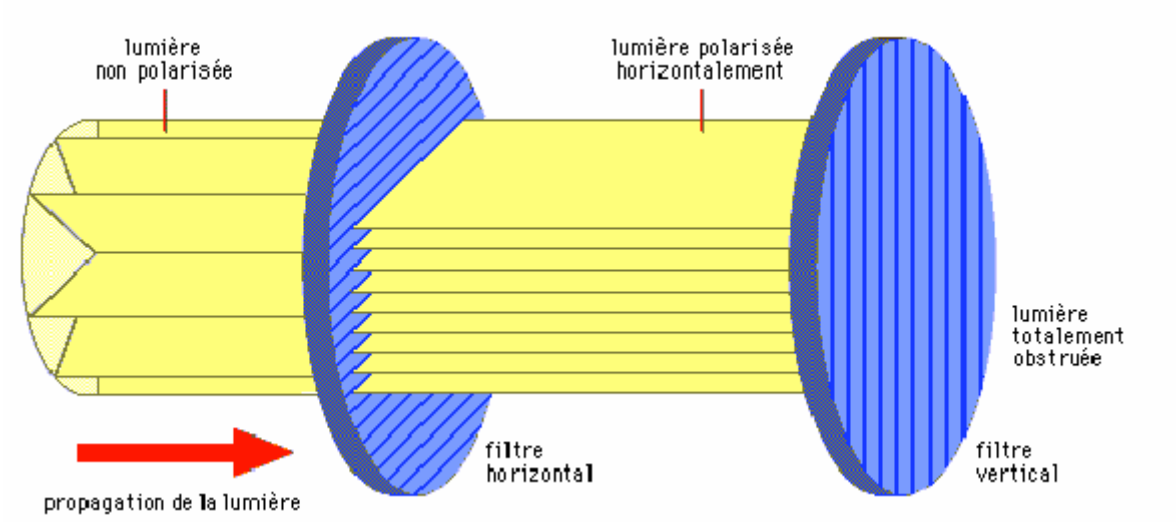
Une RAM d'affichage contient une image des tous les éléments du LCD. Chaque bit correspond à un élément du LCD. L'utilisateur doit écrire dans cette RAM à chaque fois qu'il y a modification de l'affichage.
L'utilisateur doit câbler son LCD au CI spécialisé de façon à ce que l'écriture dans la RAM d'affichage soit la plus simple possible. Par exemple si son afficheur contient des chiffres, il faut s'arranger pour que les segments d'un même chiffre correspondent à des bits consécutifs dans la mémoire d'affichage (7 bits pour les 7 segments + 1 bit pour le point décimal). Pour modifier un chiffre, il suffit d'écrire un octet dans la RAM d'affichage.

ANNEXE

PRINCIPE DE FONCTIONNEMENT D'UN AFFICHEUR À CRISTAUX LIQUIDES

1) INTRODUCTION : LUMIÈRE POLARISÉE

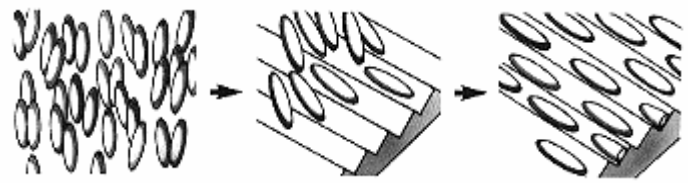
Les atomes d'une source lumineuse ordinaire émettent des impulsions de radiation d'une durée extrêmement courte. Chaque impulsion d'un seul atome est constituée d'un train d'ondes quasi monochromatique (longueur d'onde unique). Le vecteur "champ électrique" correspondant à l'onde ne tourne pas autour de la direction de propagation de cette onde mais conserve le même angle, ou azimut, avec cette direction. L'azimut initial peut prendre n'importe quelle valeur. Lorsqu'un grand nombre d'atomes émettent de la lumière, ces azimuts sont distribués au hasard ; les propriétés du faisceau lumineux sont identiques dans toutes les directions et la lumière est dite non polarisée. Si les vecteurs champs électriques de chaque onde ont le même angle azimutal (c'est-à-dire lorsque les ondes transversales se trouvent dans le même plan), la lumière est dite polarisée dans un plan ou linéairement.



En plaçant deux filtres polarisant perpendiculaires l'un de l'autre, le premier va polariser la lumière dans un sens. Le deuxième absorbe toute la lumière car celle ci arrive dans un plan perpendiculaire.
Un afficheur LCD utilise 2 filtres polarisateurs entre lesquels sont placés des cristaux liquides qui ont la propriété de permettre la rotation ou non de la lumière polarisée selon le champ électrique auquel ils sont soumis.

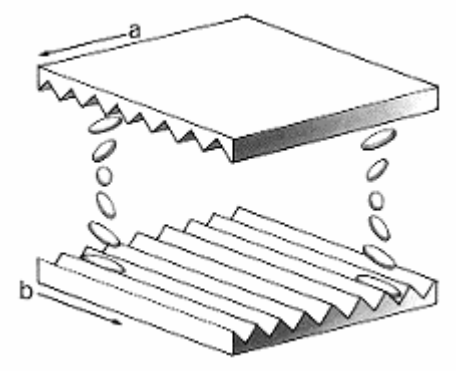
2) LES CRISTAUX LIQUIDES ET LEURS ARRANGEMENTS

Les cristaux liquides consistent en un grand nombre de cristaux allongés en suspension dans un fluide. Ces cristaux ont des propriétés d'alignement particulières.

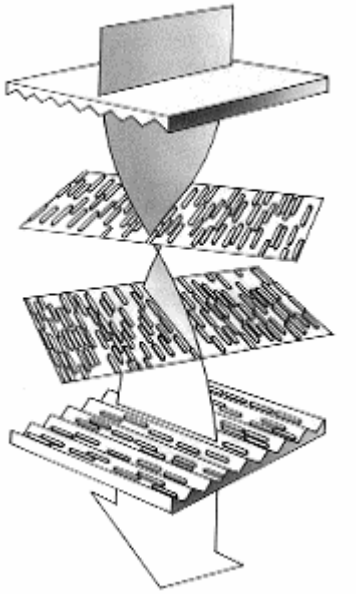


A l'état naturel, les cristaux sont en alignement dispersé	Quand les cristaux viennent en contact avec une surface finement rainurée...	ils s'alignent avec les rainures

Quand les cristaux sont placés entre 2 surfaces rainurées orthogonales, ils subissent un pivotement (twist), comme indiqué sur la figure ci-dessous.



La lumière polarisée subit elle aussi un pivotement lorsqu'elle traverse des cristaux liquides organisés comme ci-contre. Le plan de polarisation de la lumière tourne.



3) FONCTIONNEMENT ET CONSTITUTION D'UN AFFICHEUR À CRISTAUX LIQUIDES

3.1) PRINCIPE DE FONCTIONNEMENT

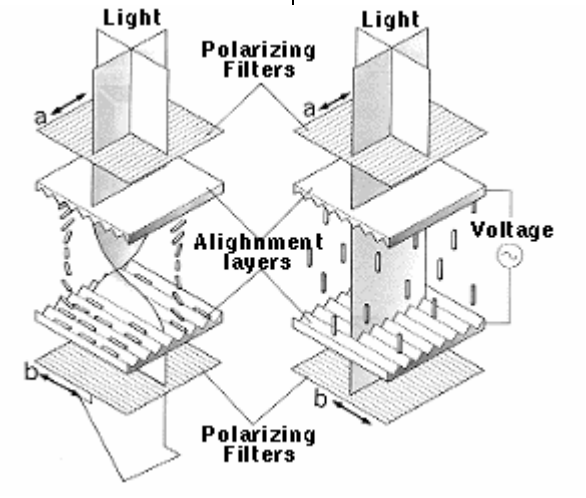
Un afficheur à cristaux liquides est constitué de :

- 2 plaques transparentes finement gravées pour l'orientation des cristaux
- électrodes conductrices transparentes déposées sur les plaques pour appliquer une tension
- cristaux liquides

Un exemple de réalisation technologique est donné plus loin.

En l'absence de champ électrique appliqué entre les électrodes, la lumière polarisée par le premier filtre subit une rotation et traverse le 2ème filtre

En présence d'un champ électrique, les cristaux changent d'alignement et le plan de polarisation de la lumière ne tourne pas entre les 2 filtres → aucun lumière ne traverse le 2ème filtre



Le principe décrit correspond au fonctionnement des afficheurs de type TN (Twisted Nematic). Il existe d'autres types d'afficheurs dans lesquels le pivotement des cristaux en l'absence de champ électrique est plus important.

3.2) AFFICHEUR PAR TRANSMISSION OU RÉFLEXION / IMAGES POSITIVE OU NÉGATIVE

Il existe 2 principes pour former une image visible par un observateur.
La source de lumière peut être :

- derrière l'afficheur (afficheur par transmission)
- la lumière ambiante (afficheur par réflexion)

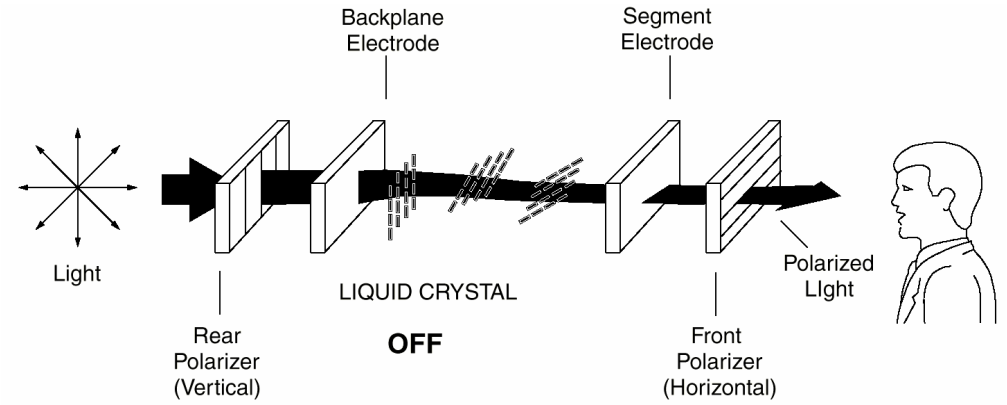
Il est possible de réaliser des images :

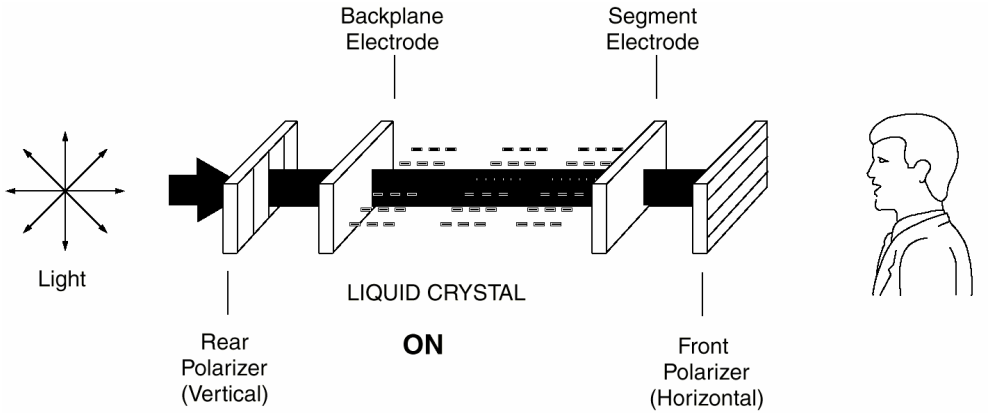
- positives : les éléments de l'afficheur apparaissent en sombre sur fond clair
- négatives : les éléments de l'afficheur apparaissent en clair sur fond sombre

Pour les images positives, de loin les plus répandues dans les LCDs utilisés dans les systèmes embarqués, un élément visible correspond à l'existence d'un champ électrique entre les électrodes.

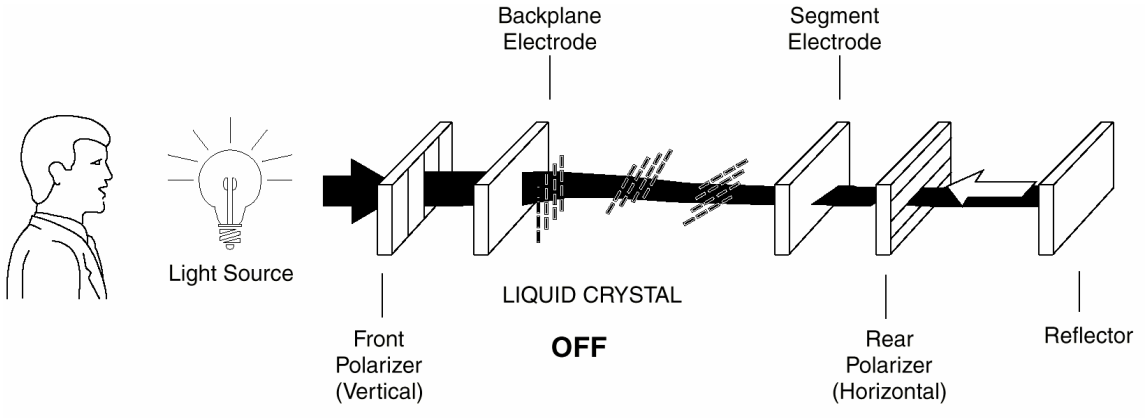
On présente ci-dessous le principe de la transmission et de la réflexion pour des images positives.

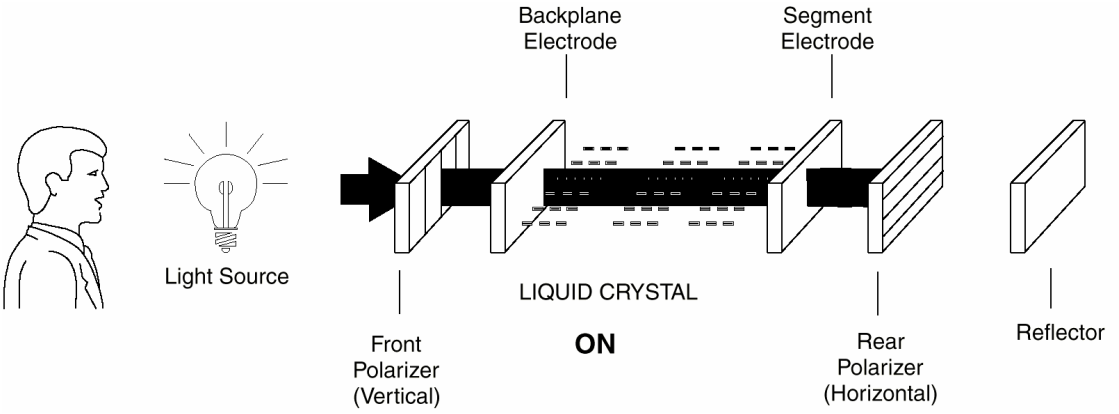
AFFICHEUR PAR TRANSMISSION





AFFICHEUR PAR RÉFLEXION



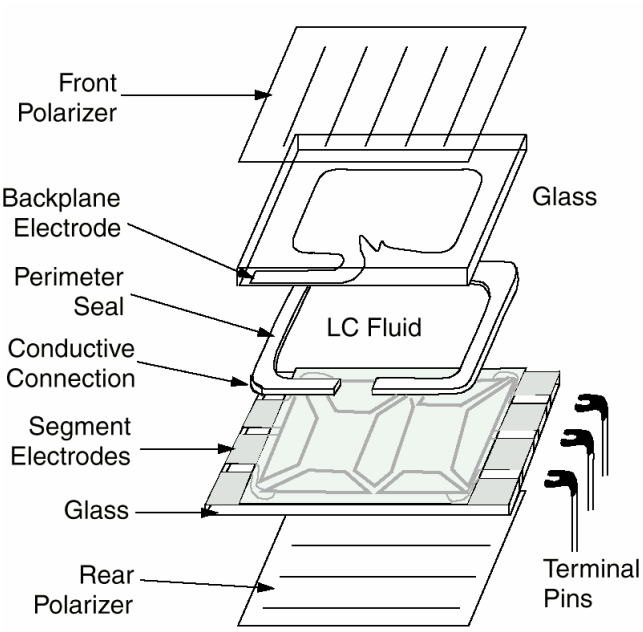


3.3) CONSTITUTION

La figure suivante montre la constitution d'un LCD.

Les rainures pour l'orientation des cristaux ne sont pas mentionnées.

L'électrode « backplane » est normalement celle qui est à l'arrière de l'afficheur.



Sources de ce document :

- Notes d'application AN658 de Microchip : LCD Fundamentals Using PIC16C92X Microcontrollers
- Doc Philips PCF8566
- Doc contrôleur Hitachi HD44780
- Document de 7 pages « Afficheurs à cristaux liquides » réalisé pour le thème industriel de terminale STI génie électronique de 98/99 (auteur non mentionné)
- Divers documents récupérés sur Internet