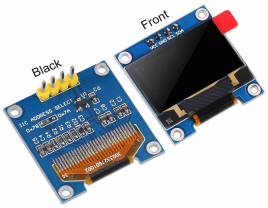
STI 2D Lazare CARNOT -ARRAS-	**Les afficheurs LCD**	SIN TP – x 3h STI2D – 2023/2024

NOM: **Prénom:**	**CLASSE: STI2D-SIN** **28/12/22**

Condition:	• travail seul ; durée x 3 heures
Matériel:	• un ordinateur sous Windows avec les logiciels Proteus et Flowcode • une carte de développement (PIC , Arduino) • un afficheur LCD
Documents:	• le sujet de cette Activité



Historique (source Wikipédia)

L'écran à cristaux liquides ou LCD (de l'anglais *liquid crystal display*) (ACL au Québec pour affichage à cristaux liquides) permet la création d'écrans plats à faible consommation d'électricité. Aujourd'hui ces écrans sont utilisés dans presque tous les affichages électroniques.

Fonctionnement de l'écran (source https://zestedesavoir.com)

Comme son nom l'indique, un écran LCD possède des cristaux liquides. Mais ce n'est pas tout ! En effet, pour fonctionner il faut plusieurs choses. Si vous regardez de très près votre écran vous pouvez voir une grille de carré. Ces carrés sont appelés des pixels (de l'anglais "Picture Element", soit "Élément d'image" en français). Chaque pixel est un cristal liquide. Lorsqu'aucun courant ne le traverse, ses molécules sont orientées dans un sens (admettons, 0°). En revanche lorsqu'un courant le traverse, ses molécules vont se tourner dans la même direction (90°). Voilà pour la base.

Composition d'un écran LCD - (CC-BY, ed g2s)

Mais pourquoi il y a de la lumière dans un cas et pas dans l'autre ?

Tout simplement parce que cette lumière est **polarisée**. Cela signifie que la lumière est orientée dans une direction. En effet, entre les cristaux liquides et la source lumineuse se trouve un filtre polariseur de lumière. Ce filtre va orienter la lumière dans une direction précise. Entre vos yeux et les cristaux se trouve un autre écran polariseur, qui est perpendiculaire au premier. Ainsi, il faut que les cristaux liquides soient dans la bonne direction pour que la lumière passe de bout en bout et revienne à vos yeux. Un schéma vaut souvent mieux qu'un long discours, je vous conseille donc de regarder celui sur la droite de l'explication pour mieux comprendre (source : Wikipédia). Enfin, vient le rétro-éclairage (fait avec des LED) qui vous permettra de lire l'écran même en pleine nuit (sinon il vous faudrait l'éclairer pour voir le contraste).

Commande du LCD

Normalement, pour pouvoir afficher des caractères sur l'écran il nous faudrait activer individuellement chaque pixel de l'écran. Un caractère est représenté par un bloc de 75 pixels. Ce qui fait qu'un écran de 16 colonnes et 2 lignes représente un total de 16275 = 1120 pixels ! Heureusement pour nous, des ingénieurs sont passés par là et nous ont simplifié la tâche.

Le décodeur de caractères

Tout comme il existe un driver vidéo pour votre carte graphique d'ordinateur, il existe un driver "LCD" pour votre afficheur. Rassurez-vous, aucun composant ne s'ajoute à votre liste d'achats puisqu'il est intégré dans votre écran. Ce composant va servir à décoder un ensemble "simple" de bits pour afficher un caractère à une position précise ou exécuter des commandes comme déplacer le curseur par exemple. Ce composant est fabriqué principalement par Hitachi et se nomme le **HC44780**. Il sert de **décodeur de caractères**. Ainsi, plutôt que de devoir multiplier les signaux pour commander les pixels un à un, il nous suffira d'envoyer des octets de commandes pour lui dire "écris moi 'STI2D CARNOT' à partir de la colonne 3 sur la ligne 1". Ce composant possède 16 broches, voici leur description :

N°	Nom	Rôle
1	VSS	Masse
2	VDD	+5V
3	VEE	Réglage du contraste
4	RS	Sélection du registre (commande ou donnée)
5	R/W	Lecture ou écriture
6	E	Entrée de validation
7 à 14	D0 à D7	Bits de données
15	A	Anode du rétroéclairage (+5V)
16	K	Cathode du rétroéclairage (masse)

Liste des broches du LCD et leur rôle

Normalement, pour tous les écrans LCD (non graphiques) ce brochage est le même. Donc pas d'inquiétude lors des branchements, sinon vous pouvez consulter le « data-sheet » (la notice constructeur) de l'écran .

Exemple : https://www.alldatasheet.fr/datasheet-pdf/pdf/146552/HITACHI/LM016L.html

Par la suite, les broches utiles qu'il faudra relier au micro-contrôleur sont les broches 4, 5 (facultatives), 6 et les données (7 à 14 pouvant être réduite à 8 à 14) en n'oubliant pas l'alimentation et la broche de réglage du contraste. Ce composant possède tout le système de traitement pour afficher les caractères. Il contient dans sa mémoire le schéma d'allumage des pixels pour afficher chacun d'entre eux. Voici la table des caractères affichables :

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Une table ASCII - (Domaine public - LanoxxthShaddow)

Quel écran choisir ? Les caractéristiques :

Texte ou Graphique ?

Dans la grande famille afficheur LCD, on distingue plusieurs catégories :

- Les afficheurs alphanumériques
- Les afficheurs graphiques monochromes
- Les afficheurs graphiques couleur

afficheur alphanumérique

afficheur graphique monochrome

afficheur graphique couleur

Les premiers sont les plus courants. Ils permettent d'afficher des lettres, des chiffres et quelques caractères spéciaux. Les caractères sont prédéfinis (voir table juste au-dessus) et on n'a donc aucunement besoin de gérer chaque pixel de l'écran. Les seconds sont déjà plus avancés. On a accès à chacun des pixels et on peut donc produire des dessins beaucoup plus évolués. Ils sont cependant légèrement plus onéreux que les premiers. Les derniers sont l'évolution des précédents, la couleur en plus (soit 3 fois plus de pixels à gérer : un sous-pixel pour le rouge, un autre pour le bleu et un dernier pour le vert, le tout forme la couleur d'un seul pixel).

Ce n'est pas la taille qui compte !

Les afficheurs existent dans de nombreuses tailles. Pour les afficheurs de type textes, on retrouve le plus fréquemment le format 2 lignes par 16 colonnes. Il en existe cependant de nombreux autres avec une seule ligne, ou 4 (ou plus) et 8 colonnes, ou 16, ou 20 ou encore plus ! Libre à vous de choisir la taille qui vous plaît le plus, pour cette activité ce sera un 2 lignes 16 colonnes.

Communication avec l'écran

La communication parallèle

De manière classique, on communique avec l'écran de manière **parallèle**. Cela signifie que l'on envoie des bits **par blocs**, en utilisant plusieurs broches en même temps (opposée à une transmission série où les bits sont envoyés un par un sur une seule broche). Comme expliqué plus tôt dans ce chapitre, nous utilisons 10 broches différentes, 8 pour les données (en parallèle donc) et 2 pour de la commande (E : Enable et RS : Registre Selector). La ligne R/$\overline{W}$ doit être connectée à la masse (VSS) si l'on souhaite uniquement faire de l'écriture.

Pour envoyer des données sur l'écran, c'est en fait assez simple. Il suffit de suivre un ordre logique et un certain timing pour que tout se passe bien. Tout d'abord, il nous faut placer la broche RS à 1 ou 0 selon que l'on veut envoyer une commande, par exemple "déplacer le curseur à la position (1;1)" ou que l'on veut envoyer une donnée : "écris le caractère 'a' ". Ensuite, on place sur les 8 broches de données (D_7 à D_0) la valeur de la donnée à afficher. Enfin, il suffit de faire une impulsion d'au moins 450 ns pour indiquer à l'écran que les données sont prêtes. C'est aussi simple que ça !

Cependant, comme les ingénieurs d'écrans sont conscients que la communication parallèle prend beaucoup de broches, ils ont inventé un autre mode que j'appellerai "semi-parallèle". Ce dernier se contente de travailler avec seulement les broches de données D_7 à D_4 (en plus de RS et E) et il faudra mettre les quatre autres (D_3 à D_0) à la masse. Il libère donc quatre broches. Dans ce mode, on fera donc deux fois le cycle "envoi des données puis impulsion sur E" pour envoyer un octet complet.

La communication série

Lorsque l'on ne possède que très peu de broches disponibles sur notre Arduino, il peut être intéressant de faire appel à un composant permettant de communiquer par voie série avec l'écran. Un tel composant se chargera de faire la conversion entre les données envoyées sur la voie série et ce qu'il faut afficher sur l'écran. Le gros avantage de cette solution est qu'elle nécessite seulement un seul fil de donnée (avec une masse et le VCC) pour fonctionner là où les autres méthodes ont besoin de presque une dizaine de broches.

Nous resterons pour notre activité dans le classique en utilisant une connexion parallèle. En effet, elle nous permet de garder l'approche "standard" de l'écran et nous permet de garder la liaison série pour autre chose.

La communication par liaison I²C

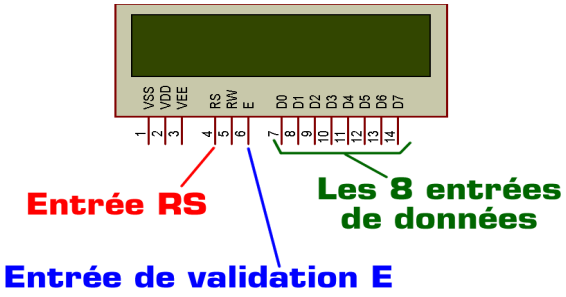
Un dernier point à voir, c'est la communication d'un microcontrôleur vers l'écran par la liaison I²C. Cette liaison est utilisable avec seulement 2 broches (une broche de donnée et une broche d'horloge) et nécessite l'utilisation de deux broches analogiques. Mais cela fera partie d'une autre activité.

✍ Passons à l'activité en classe :

Pour celle-ci on se servira d'afficheur de la première catégorie, car ils suffisent à faire de nombreux montages et restent accessibles financièrement. L'objectif de cette activité consiste en la mise en œuvre de cet écran, sa configuration et utilisation des lignes de port (entrées/sorties) d'un microcontrôleur.

Rappels : Un afficheur LCD contient :

- une entrée de contrôle **RS** (Register Select)
- une entrée de validation **E** (Enable)
- 8 entrées de données **D7** à **D0**



RS permet de préciser si la commande présente sur les entrées **D7** à **D0** , est une **instruction** ou une **donnée**,
E permet de valider cette commande.

Une commande est une valeur numérique présente sur les entrées **D7** à **D0** et validée par une impulsion sur **E**.

Le protocole d'envoi des commandes précise la liste des instructions à envoyer pour configurer l'afficheur (**RS=0**) suivie des données à envoyer (**RS=1**).

L'entrée **R/$\overline{W}$** sera mise à zéro (connectée à la masse) et sera inutilisée ici.

Les 3 broches d'alimentation **VSS, VDD** et **VEE** n'ayant pas besoin d'être obligatoirement alimentées dans ISIS Proteus, elles resteront non connectées dans les exemples ci-dessous.

La seule question qui nous intéresse ici est "***Que faut-il mettre sur les entrées de données et quel protocole faut-il utiliser pour programmer l'afficheur ?***".

1re Partie : Programmation d'un afficheur LCD pas à pas en mode 8 bits :

En mode 8 bits les 8 entrées D_7 à D_0 sont câblées et les commandes sont sur 8 bits (1 octet).

Il faut envoyer à l'afficheur 3 octets d'initialisation afin de le configurer avant d'envoyer les caractères à afficher.

Envoi des instructions de configuration			Envoi des données pour afficher les caractères
RS=0			RS=1
$38	$0E	$06	on envoie le code ASCII de chaque caractère

Rôle de chacun des 3 octets de configuration :

$38 : configure l'afficheur en mode 8 bits avec 2 lignes

$0E : configure le curseur visible (à remplacer par **$0C** si on veut le curseur invisible ou par **$0F** pour un curseur clignotant)

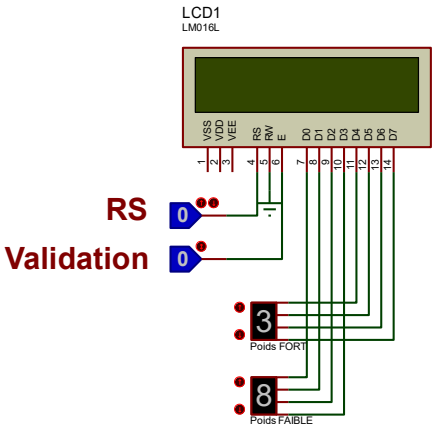
$06 : configure le déplacement du curseur vers la droite

A vous de faire :

1) Dans un premier on vous demande de réaliser le schéma structurel sous Isis Proteus ci dessous permettant le pilotage manuel d'un afficheur LCD LM016L en mode 8 bits.

   Liste des composants nécessaire sous proteus :

Nom réel du composant	Nom exact du composant (colonne Device dans la catégorie) à saisir dans Mots clés	Catégorie contenant le composant
Afficheur LCD alphanumérique 16x2	LM016L	Optoelectronics
Simulateur d'états logique « action verrouillée »	LOGICSTATE	Debugging tools
Simulateur d'états logique « action momentanée »	LOGICTOGGLE	Debugging tools
Interrupteur interactif hexadécimal (sortie 4 bits)	THUMBSWITCH-HEX	Switches and Relays



2) Lancer la simulation puis réalisez manuellement la séquence permettant d'afficher le caractère suivant : @

2e Partie : Programmation d'un afficheur LCD pas à pas en mode 4 bits :

En mode 4 bits seules les 4 entrées **D_7** à **D_4** sont câblées et les commandes sont sur 4 bits (1 quartet).

En mode 4 bits les instructions comme les codes ASCII des caractères sont envoyés en 2 étapes :

le quartet de **poids fort** en premier, puis ensuite le quartet de **poids faible**.

Pour configurer l'afficheur en 4 bits il faut commencer par lui envoyer les 4 quartets **$3 $3 $3** et **$2**.

Envoi des instructions de configuration								**Envoi des données pour afficher les caractères**
RS=0								**RS=1**
$3	**$3**	**$3**	**$2**	**$0**	**$E**	**$0**	**$6**	**on envoie le code ASCII de chaque caractère en 2 étapes**

L'instruction **$0E** affichant le curseur est envoyée ici en 2 quartets : **$0** puis **$E**

L'instruction **$06** configurant le déplacement du curseur vers la droite est envoyée ici en 2 quartets : **$0** puis **$6**

Une fois l'afficheur configuré en mode 4 bits, les codes ASCII des caractères à afficher sont envoyés en 2 étapes : le quartet de **poids fort** en premier, puis ensuite le quartet de **poids faible**.

1) Proposez un schéma structurel permettant d'utiliser l'afficheur en mode 4 bits puis faire valider celui-ci par votre professeur.
2) Rappelez pourquoi dans certain cas ce mode est utilisé.
3) Lancer la simulation puis réalisez manuellement la séquence permettant d'afficher le caractère suivant : &

3e Partie : Affichage d'un ensemble de caractères en mode 8 Bits :

Dans cette partie de l'activité on vous propose d'afficher, cette fois-ci, une suite de caractères.

Voici donc le message à afficher :

Carnot-ARRAS

SIN

Vous présenterez, à la suite de votre contre rendu, votre démarche ainsi que les informations qui sont envoyés sur les ports D_7 à D_0, RS, R/$\overline{W}$ et E pour chaque étapes intermédiaires avant l'obtention du résultat final.

Quelques instructions bien pratiques

Voici les instructions les plus utilisées pour chacun des afficheurs LCD utilisables dans ISIS Proteus :

Instructions valables pour tous les afficheurs LCD	
Action à réaliser	Valeur de l'instruction (à envoyer avec RS=0)
Curseur invisible	$0C
Curseur visible en fixe	$0E
Curseur visible clignotant	$0F
Déplacement du curseur vers la droite	$06
Effacer l'écran	$01
Bascule l'afficheur en mode 4 bits	$33 $32
Bascule l'afficheur en mode 8 bits	$33 $33
Configure l'afficheur en multi-lignes en 4 bits	$28
Configure l'afficheur en multi-lignes en 8 bits	$38

Séquence d'initialisation complète pour configurer un afficheur LCD en mode 8 bits	
Action à réaliser	Valeur de l'instruction (à envoyer avec RS=0)
Bascule l'afficheur en mode 8 bits	$33 $33
Configure l'afficheur en multi-lignes en 8 bits	$38
Affichage du curseur	$0E
Déplacement du curseur vers la droite	$06
Ensuite on peut envoyer les codes ASCII des caractères à afficher octet par octet	

Séquence d'initialisation complète pour configurer un afficheur LCD en mode 4 bits	
Action à réaliser	Valeur de l'instruction (à envoyer avec RS=0)
Bascule l'afficheur en mode 4 bits	$33 $32
Configure l'afficheur en multi-lignes en 4 bits	$28
Affichage du curseur	$0E
Déplacement du curseur vers la droite	$06
Ensuite on peut envoyer les codes ASCII des caractères à afficher quartet par quartet (avec le poids fort en premier)	

Changement de ligne sur les afficheurs possédant 2 lignes :

Instructions valables pour les afficheurs LM016L (2x16) LM032L (2x20) LM017L (2x32) et LM018L (2x40)	
Action à réaliser	Valeur de l'instruction (à envoyer avec RS=0)
Déplacer le curseur au début de la ligne 1	$80
Déplacer le curseur au début de la ligne 2	$C0

Changement de ligne sur les afficheurs possédant 4 lignes :

Instructions valables pour l'afficheur LM041L (4x16)	
Action à réaliser	Valeur de l'instruction (à envoyer avec RS=0)
Déplacer le curseur au début de la ligne 1	$80
Déplacer le curseur au début de la ligne 2	$C0
Déplacer le curseur au début de la ligne 3	$90
Déplacer le curseur au début de la ligne 4	$D0

Instructions valables pour l'afficheur LM044L (4x20)	
Action à réaliser	Valeur de l'instruction (à envoyer avec RS=0)
Déplacer le curseur au début de la ligne 1	$80
Déplacer le curseur au début de la ligne 2	$C0
Déplacer le curseur au début de la ligne 3	$94
Déplacer le curseur au début de la ligne 4	$D4

4e Partie : Pour aller plus loin: Utilisation de Flowcode et d'Isis Proteus.

Maintenant que vous avez effectué toutes ces manipulations fastidieuses dues à la programmation manuel, nous allons utiliser les outils disponibles en classe afin de faciliter notre tâche.

1) Proposez, en vous aidant de l'activité « Gestion d'un Feux Tricolores Mode Automatique » le schéma structurel, avec une carte Arduino Uno permettant de répondre à notre besoin.
2) Proposez ensuite sous Flowcode l'algorithme permettant de répondre à la troisième partie de cette activité.
3) Testez votre programme sous Flowcode puis en simulation sous Isis Proteus.
4) Enfin pour terminer, câblez et testez votre montage.

Voilà !!! c'est fini...pour le moment. N'hésitez pas à partager avec votre professeur par l'intermédiaire de votre compte rendu