



LCD

PROGRAMMER SA CGRAM

Ce document explique le principe de fonctionnement de la CGRAM d'un afficheur LCD.

Et j'insiste bien sur le mot : "*principe*".

Car il n'est pas question ici de détailler techniquement le fonctionnement exact de l'afficheur, savoir si les données sont sur 5, 7 ou 8 bits. On s'en fiche un peu, et à ce sujet la documentation disponible sur le Net est très abondante et compréhensible.

L'important est de savoir comment fonctionne la CGRAM, connaître son principe de fonctionnement afin de pouvoir, très simplement vous allez le constater, créer et afficher ses propres caractères.

Les exemples se basent sur un afficheur LCD 2 lignes 16 caractères et avec une taille de fonte de caractères de 5 points x 7 points.

Mais le principe de fonctionnement est strictement identique pour (presque) tous les autres modèles d'afficheurs.

Les très courts exemples de programmation présentés dans ce document sont écrits en langage assembleur (PIC).

Mais adaptables sans difficulté à tous micros et tous langages.

Vous êtes prêt à commencer ?

Oui ? Alors c'est parti...

LA CGROM

C'est la mémoire qui contient les caractères qui vont pouvoir être copiés dans la DDRAM et donc pouvoir être affichés sur l'écran.

Son contenu :

Higher 4bit Lower 4bit	0000	0010	0011	0100	0101	0110	0111	1010	1011	1100	1101	1110	1111
xxxx0000		0	@P`P					-	9	3	ap		
xxxx0001		!	1AQa9										
xxxx0010		"	2ERbr										
xxxx0011		#	3CScs										
xxxx0100		\$	4DTdt										
xxxx0101		%	5EUeu										
xxxx0110		&	6FUfu										
xxxx0111		'	7GWgw										
xxxx1000		(8HXhx										
xxxx1001)	9IYiy										
xxxx1010		*	: JZjz										
xxxx1011		+	: KLk<										
xxxx1100		,	<L#ll										
xxxx1101		-	=Mln>										
xxxx1110		.	>N^n+										
xxxx1111		/	?O_o+										

Nous avons tous les caractères couramment employés sauf malheureusement les caractères accentués.

Mais nous allons pouvoir contourner ce manque grâce à la CGRAM.

Ok, j'ai tout compris !

Mais maintenant que dois-je faire pour justement pouvoir afficher ce que je veux ?...

LES SIGNAUX DE CONTRÔLE

Le LCD dispose de trois entrées permettant de définir les modes de communication.

Le signal R/W : indique au LCD si le mot présent sur son bus de données (4 ou 8 bits) est en lecture ou en écriture :

R/W = 0 => écriture

R/W = 1 => lecture

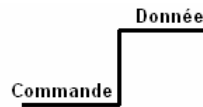
Le signal RS : indique au LCD si le mot présent sur son bus de données est une commande ou une donnée.

Le LCD possède en interne deux registres : un registre d'instruction (commande) et un autre de données.

La sélection d'un de ces deux registres s'effectue par la mise au niveau correspondant du signal RS :

RS = 1 => donnée

RS = 0 => commande



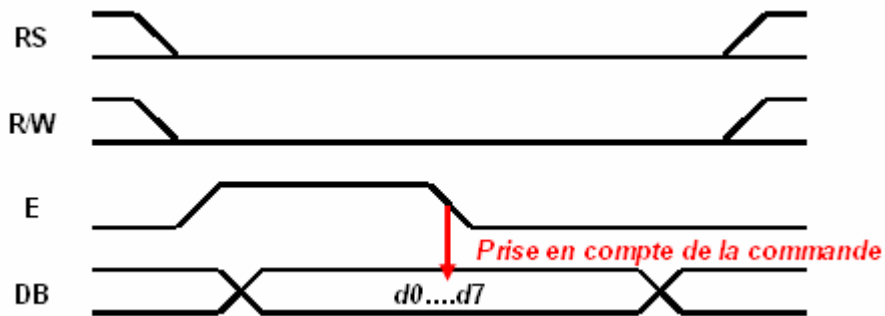
Nous sélectionnons le registre d'instruction lorsque nous voulons envoyer une commande : effacement de l'afficheur (cls), positionnement du curseur, curseur invisible, clignotant, etc...

Et le registre de données lorsque nous voulons afficher un caractère par exemple.

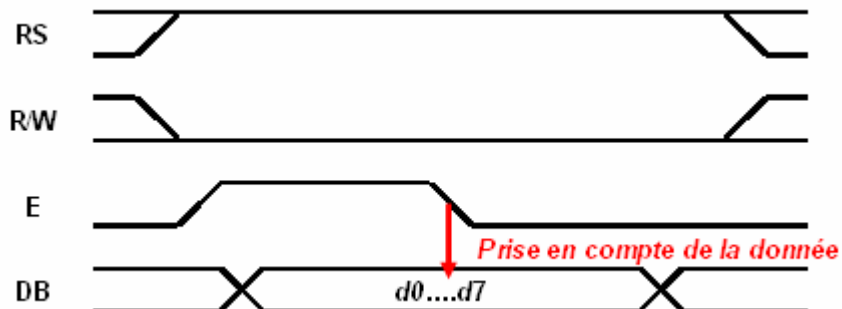
Le signal E : c'est le signal d'horloge du LCD.

Indique au LCD, lorsque vous faites passer ce signal du niveau haut au niveau bas, que les données présentes sur son bus sont valides.

Envoi d'une commande (RS = 0) :



Envoi d'une donnée (RS=1) :



Dans les exemples cités, nous utilisons deux sous-programmes pour envoyer les informations au LCD :

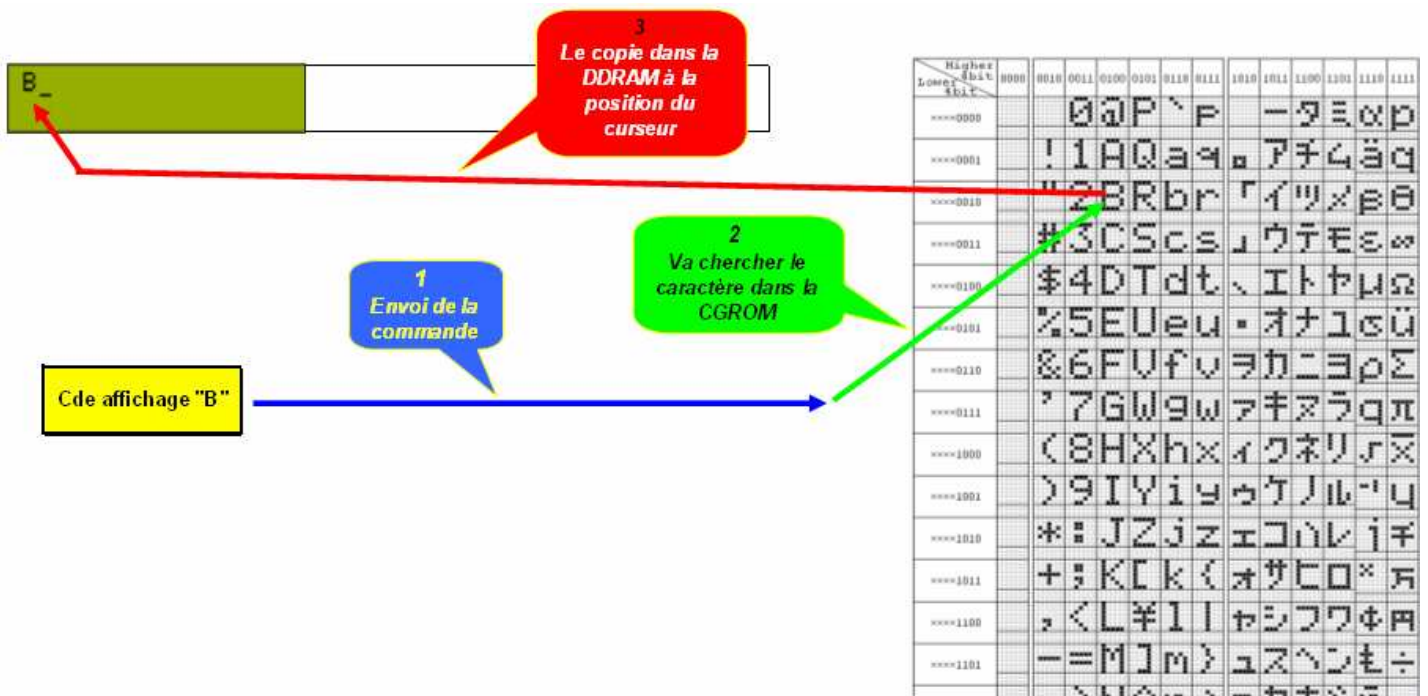
```
call lcd_car      ;envoi d'une donnée contenue dans w (avec RS = 1 et RW = 0)
call lcd_cde     ;envoi d'une commande contenue dans w (avec RS = 0 et RW = 0)
```

Ces deux sous routines ne seront pas détaillés ici car dépendant de l'affectation des ports connectés au LCD (donc propre à votre schématique).

Comme indiqué en préalable, il n'est expliqué dans ce document que le principe de fonctionnement. Pour tout ce qui traite de l'initialisation du LCD, de l'envoi des différentes commandes etc... se reporter à la volumineuse documentation disponible sur le Net.

POUR AFFICHER UN CARACTÈRE QUE DOIS-JE FAIRE ET QUE SE PASSE-T-IL ?

D'abord ce qui se passe :



1/ Vous envoyez votre demande d'affichage (par exemple la lettre "B")

2/ Le LCD va chercher dans la CGROM le caractère demandé

3/ Et le copie dans la DDRAM à la position (emplacement) du curseur

Et ce que je dois faire :

```
movlw 'B'          ;w= code ascii de "B" (0x42)
call  lcd_car      ;Envoi de (w) vers afficheur
```

Elle est pas belle la vie ?

Nous allons maintenant utiliser la CGRAM.

Pour cela il est préférable de connaître le principe de fonctionnement interne du LCD.

Nous avons notre CGROM, mémoire qui contient les caractères qui vont pouvoir être copiés dans la DDRAM et donc affichés sur l'écran.

Mémoire ROM, c'est-à-dire figée et non modifiable.

Higher Lower 4bit 4bit	0000	0010	0011	0100	0101	0110	0111	1010	1011	1100	1101	1110	1111
xxxx0000		0	1	2	3	4	5	6	7	8	9	A	B
xxxx0001		C	D	E	F	G	H	I	J	K	L	M	N
xxxx0010		O	P	Q	R	S	T	U	V	W	X	Y	Z
xxxx0011		[\]	^	_	`	{		}	~		
xxxx0100													
xxxx0101													
xxxx0110													
xxxx0111													
xxxx1000													
xxxx1001													
xxxx1010													
xxxx1011													
xxxx1100													
xxxx1101													
xxxx1110													
xxxx1111													

Que de la ROM ?

Pas tout à fait...

C'est pour cela, afin de comprendre plus facilement le principe de fonctionnement, que j'ai inclus la CGRAM dans la CGROM (et elle en fait partie d'ailleurs).

En effet, la CGROM comprend deux parties :

La partie en ROM (encadrée en bleu) dans laquelle sont programmés (en usine) les différents caractères que nous connaissons.

De l'adresse 0x20 à 0xFF.

Nous ne pouvons rien modifier.

Puis notre CGRAM (encadré en rouge), zone RAM dans laquelle nous allons pouvoir écrire nos propres caractères.

De l'adresse 0x00 à 0x07 (huit caractères uniquement).

La CGROM comprend donc la CGRAM et la ROM.

Pour afficher un caractère sur l'écran, vous savez tous comment faire, il suffit d'envoyer le caractère désiré à l'afficheur.

Par exemple, si nous voulons afficher la lettre "A", nous exécutons les instructions :

```
movlw 'A'           ;w= code ascii de "A" (0x41)
call  lcd_car       ;Envoi de (w) vers afficheur
```

Normal.

On envoi "A" et il affiche "A".

Hé bien non ! Pas forcément. Car ce n'est pas tout à fait comme cela que ça fonctionne...

En réalité, lorsque vous envoyez le code 0x41h à l'afficheur, contrairement à ce que vous pouvez penser, vous ne lui demandez pas d'afficher la lettre "A".

Vous n'indiquez au LCD qu'une **adresse** dans la CGROM, qui contient le caractère que vous voulez afficher.

Et uniquement l'adresse dans la CGROM, pas le code ascii en lui-même.

A la réception de cette adresse, le LCD copie dans la DDRAM le caractère qui est contenu dans la CGROM à cette adresse.

Alors bien sûr, lors de la programmation de la ROM en usine, le fabricant a fait correspondre les adresses avec les codes ascii.

Et ça nous arrange !

Mais un industriel peut parfaitement demander au fabricant une programmation personnalisée de la ROM. Et dans cas, à l'adresse 0x41h, il n'est certain qu'il y ait la lettre "A".

Higher Lower 4bit 4bit	0000	0010	0011	0100	0101	0110	0111	1010	1011	1100	1101	1110	1111
xxxx0000	A	B	C	D	E	F	G	H	I	J	K	L	M
xxxx0001 01h	S	!	1	A	Q	a	9	u	7	7	4	3	q
xxxx0010	"	2	B	R	b	r	Γ	ι	υ	×	β	θ	
xxxx0011	#	3	C	S	c	s	┘	ο	τ	ε	σ	ω	
xxxx0100	\$	4	D	T	d	t	√	ι	κ	ρ	μ	α	
xxxx0101	%	5	E	U	e	u	•	才	大	1	ε	ü	
xxxx0110	&	6	F	V	f	v	ヲ	カ	ニ	ヨ	ρ	Σ	
xxxx0111	'	7	G	W	g	w	ア	チ	又	ヲ	q	π	

Si nous exécutons les instructions :

```
movlw 0x53
call lcd_car
```

La lettre "S" va s'afficher à la position du curseur dans la DDRAM car dans la CGROM, à l'adresse 0x53, se trouve le caractère "S".

Mais supposons que nous ayons programmé notre CGRAM avec, à l'adresse 0x01, la lettre "S".

Nous pouvons aussi afficher ce "S" en exécutant :

```
movlw 0x01
call lcd_car
```

Car notre 0x01 n'est pas le caractère lui-même, mais uniquement l'adresse du caractère se trouvant dans la CGROM à cette adresse (0x01 = donc dans la partie CGRAM).

Alors tout est simple, il ne nous reste plus qu'à programmer la CGRAM avec NOS caractères.

Jusqu'à huit caractères définis entièrement par nous et mémorisés de l'adresse 00h à 07h.

Ensuite, pour afficher un de ces caractères nous n'aurons qu'à exécuter les instructions :

```
movlw xx           ;w = xx = adresse du caractère dans la CGROM
                   ;(de 00h à 07h)
call lcd_car       ;Envoi de (w) vers afficheur
```

ACCÉDER À LA CGRAM

Le LCD possède deux blocs de RAM différents :

1/ La DDRAM = ce qui peut être affiché sur l'écran (voir page 2)

2/ Et la CGRAM = incluse dans la CGROM et qui contient nos propres caractères

Par défaut (au démarrage) le LCD pointe sur la DDRAM.

Dans ce cas, toute donnée envoyée au LCD est considérée comme une demande d'affichage, donc une copie immédiate du caractère de la CGROM vers la DDRAM.

Afin de pouvoir "travailler" sur la CGRAM, pour programmer nos caractères, il va falloir l'indiquer au LCD.

Nous le faisons en envoyant la commande *Set CG RAM address* :

```
movlw B'01xxxxxx'      ; w=xxxxxx = adresse dans la CGRAM (de 00h à 3Fh)
call  lcd_cde           ;Envoi vers afficheur (c'est une commande : RS=0 et RW=0)
                          Adresse de la ligne dans la CGRAM (00h à 3Fh)? Explications plus loin...
```

Dans ce cas nous travaillons sur la CGRAM, et toute donnée envoyée est considérée par le LCD comme une commande d'écriture dans la CGRAM (ce qui nous permet donc de définir nos caractères).

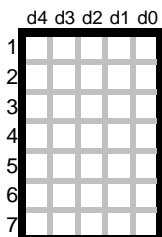
Il n'y a dans ce cas aucun affichage.

Une fois fini, et pour revenir sur la DDRAM, nous devons envoyer la commande *Set DD RAM address* :

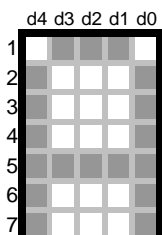
```
movlw B'1xxxxxxx'      ;w= xxxxxxxx = adresse dans la DDRAM (de 00h à 7Fh)
call  lcd_cde           ;Envoi vers afficheur (c'est une commande : RS=0 et RW=0)
```

Voyons comment sont structurés ces caractères :

Comme indiqué précédemment tous les exemples de ce document se basent sur une taille de fonte de caractères de 5 points x 7 points.



Chaque caractère est donc composé de 7 lignes de 5 pixels chacune.



	d4	d3	d2	d1	d0
1	0	1	1	1	0
2	1	0	0	0	1
3	1	0	0	0	1
4	1	0	0	0	1
5	1	1	1	1	1
6	1	0	0	0	1
7	1	0	0	0	1

Pour chaque ligne, chacun des pixels peut prendre la valeur 1 (point allumé sur l'écran) ou 0 (point éteint sur l'écran).

Pour la lettre "A" par exemple.

Nous pouvons donc définir nos propres caractères en positionnant les pixels désirés à 1.

La lettre accentuée "é" par exemple :

	d4	d3	d2	d1	d0		d4	d3	d2	d1	d0
1						0	0	0	1	0	
2						0	0	1	0	0	
3						0	1	1	1	0	
4						1	0	0	0	1	
5						1	1	1	1	1	
6						1	0	0	0	0	
7						0	1	1	1	0	

Nous envoyons la commande *Set CG RAM address* (pour travailler avec la CGRAM) :

```
movlw B'01000000'    ;Nous pointons sur la 1ère ligne du 1er caractère (adresse 0x00)
                      ;de la CGRAM
call  lcd_cde        ;Envoi de la commande vers afficheur
```

Puis nous envoyons à l'afficheur les codes correspondants aux pixels allumés/éteints pour chaque ligne de ce caractère "é".

Seuls les cinq bits de poids faibles sont significatifs.

```
movlw B'00000010'    ;1ère ligne
call  lcd_car
movlw B'00000100'    ;2ème ligne
call  lcd_car
movlw B'00001110'    ;3ème ligne
call  lcd_car
movlw B'00010001'    ;4ème ligne
call  lcd_car
movlw B'00011111'    ;5ème ligne
call  lcd_car
movlw B'00010000'    ;6ème ligne
call  lcd_car
movlw B'00001110'    ;7ème ligne
call  lcd_car
movlw B'00000000'    ;Obligatoire : toujours envoyer 8 lignes (la 8ème est donc à 0)
call  lcd_car
```

Pour l'exemple et pour facilité de compréhension, le mode de programmation présenté est très "linéaire" !

Si vous en avez plusieurs caractères à définir, vous continuez à envoyer les lignes correspondantes du 2^{ème} caractère... jusqu'au 8^{ème} éventuellement (8 lignes pour chaque caractère).

Nous avons donc programmé notre premier caractère à l'adresse 0x00 dans la CGROM.

Pour l'afficher, après être revenu à la DDRAM par la commande *Set DD RAM address*, il suffit d'exécuter :

```
movlw 0x00          ;w = 00 = adresse du caractère "é" dans la CGRAM
call  lcd_car       ;Envoi de (w) vers afficheur
```

Et nous obtenons :



Nous sommes malheureusement limité à huit caractères.

C'est parfois insuffisant.

Mais, au cours de l'exécution de votre programme, vous pouvez à tout moment changer tout ou partie du contenu de la CGRAM.

Ne modifier même qu'un seul caractère.

Dans ce cas, attention à l'adressage lorsque vous programmez votre caractère car cet adressage ne doit pas indiquer le numéro du caractère (0 à 7) mais le numéro de la 1ère ligne du caractère dans la CGRAM.

Et il y a 8 lignes pour chaque caractère.

	<i>Lignes de</i>
Caractère adresse 0	0 à 7
Caractère adresse 1	8 à 15
Caractère adresse 2	16 à 23
Caractère adresse 3	24 à 31
Caractère adresse 4	32 à 39
Caractère adresse 5	40 à 47
Caractère adresse 6	48 à 55
Caractère adresse 7	56 à 63

Supposons par exemple que vous ne vouliez ne modifier que le 4^{ème} caractère dans la CGRAM :

En premier, envoi de la commande *Set CG RAM address* (pour pointer sur la CGRAM) :

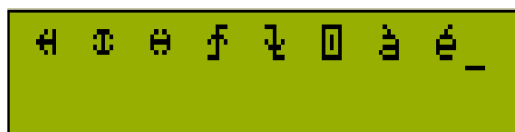
```
movlw B'01011000'      ;(58h) – avec "11000"= 24 qui pointe donc la 24ème ligne de  
                        ;la CGRAM (1ère ligne du 4ème caractère – adresse 03)  
  
call  lcd_cde           ;Envoi de la commande vers afficheur
```

Puis les 8 lignes définissant votre caractère :

```
movlw 1ère ligne  
call  lcd_car  
movlw 2ème ligne  
call  lcd_car  
...  
...  
...  
movlw 7ème ligne  
call  lcd_car  
  
movlw 0x00              ;Toujours 8 lignes donc la 8ème à 0  
call  lcd_car
```

Une fois terminé ne pas oublier d'exécuter la commande *Set DD RAM address* pour revenir à la DDRAM !

Et il n'y a pas que les caractères accentués. Vos besoins et votre imagination feront le reste :

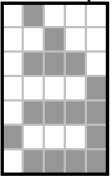


Afin de vous aider à "programmer" votre CGRAM, le fichier Excel "**CGRAM.xls**" vous permet d'obtenir directement et rapidement les octets à envoyer à votre LCD :

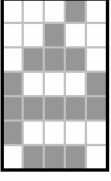
Copie d'écran

**Tapez un "x" (sans guillemets) dans chaque cellule où vous voulez que le pixel soit allumé sur l'écran
Les huit octets à envoyer au LCD s'affichent automatiquement**

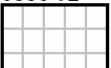
Adresse 00 (CGRAM)
08h,04h,0eh,01h,0fh,11h,0fh,00h



Adresse 01
02h,04h,0eh,11h,1fh,10h,0eh,00h



Adresse 02
00h,00h,00h,00h,00h,00h,00h,00h



Ce document a pour but de mieux faire connaître le fonctionnement et l'utilisation de la CGRAM.

De ce fait, nombre d'autres détails techniques concernant toutes les possibilités (dont la programmation) du LCD ont été occultés.

Mais pour combler ce manque, il vous suffit de taper par exemple "LCD" sur Google (ou sur votre moteur de recherche favori) !

Sous réserve d'erreurs ou omissions...

Le 19 octobre 2010

Asl

Droits d'utilisation

Le présent document peut être librement diffusé, mais toujours dans son intégralité.

Tous les droits sur le contenu de ce document, textes et schémas qui l'accompagnent, demeurent la propriété exclusive de **Génération Hydrogène**.

De ce fait, toute reproduction partielle est strictement interdite.

L'auteur ne pourra être tenu pour responsable d'aucune conséquence directe ou indirecte résultant de la lecture et/ou de l'application décrite dans le présent document.

Toute utilisation commerciale est interdite sans l'accord express de l'administrateur de **Génération Hydrogène**.