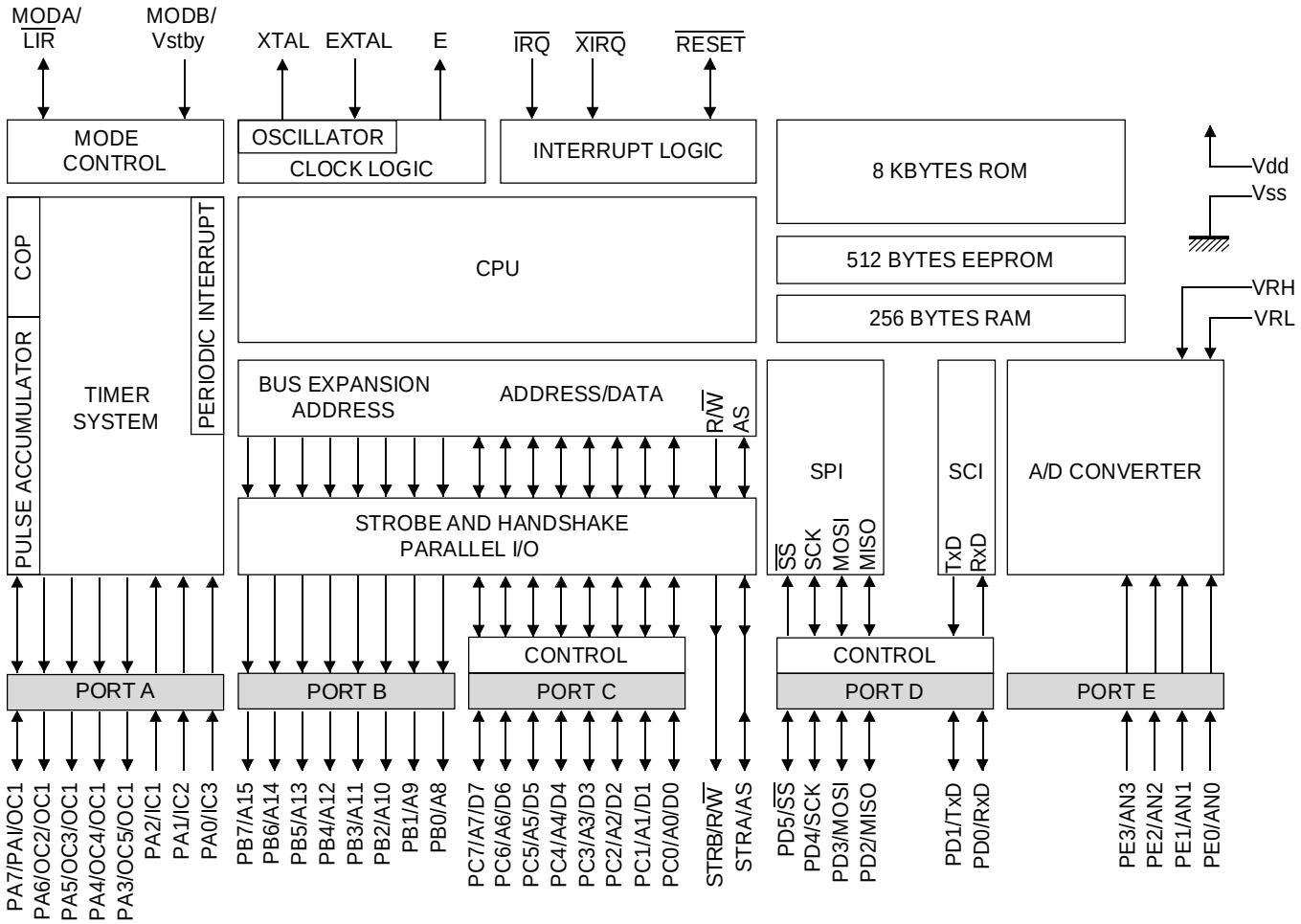
Etude de l'afficheur et du programme d'affichage

afficheur LCD: liquid crystal display

Rappel sur le microcontrôleur 68hc11

Les Ports d'entrées/sorties du 68hc11



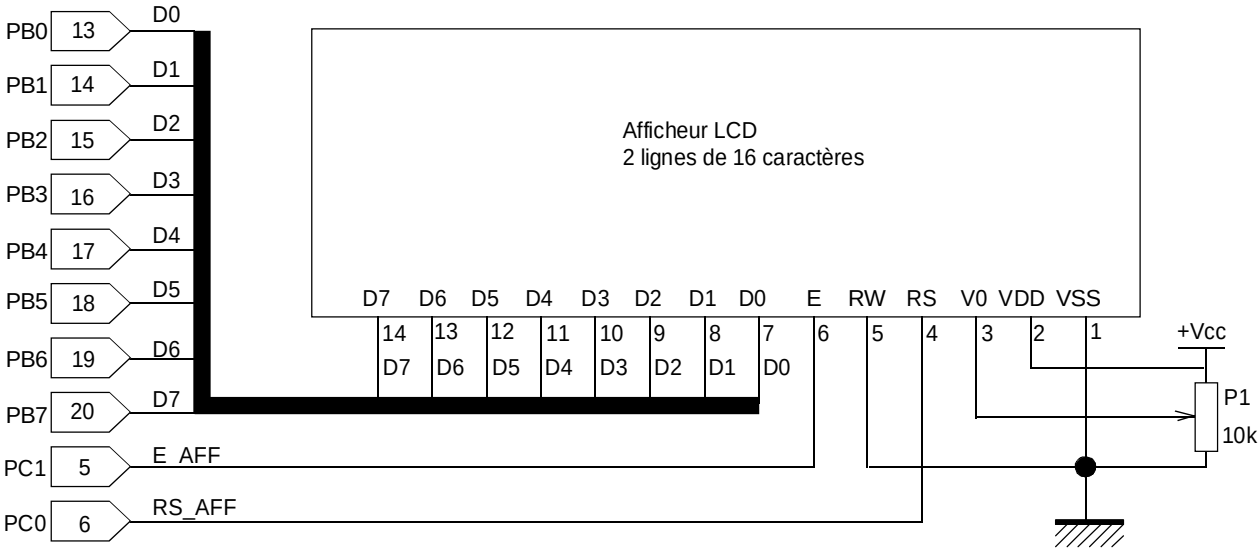
Les Ports du 68hc11 sont soit unidirectionnels en entrée ou en sortie, soit bidirectionnels auquel cas ils doivent être configurés par l'utilisateur.

Port	Adresse	Bidirectionnel	adresse du registre de direction
Port A	$1000	PA7	B7 de PACTL $1026
Port B	$1004	en sortie	
Port C	$1003	oui	DDRC = $1007
Port D	$1008	oui	DDRD = $1009
Port E	$100A	en entrée	

DDR: ***D***ata ***D***irection ***R***egister

inaccessible sur le kit MCU11

Interconnexion entre kit et afficheur (carte afficheur - clavier)



connecteur J1 du kit	afficheur LCD
1- PB0 ($1004)	D0
2- PB1 ($1004)	D1
3- PB2 ($1004)	D2
4- PB3 ($1004)	D3
5- PB4 ($1004)	D4
6- PB5 ($1004)	D5
7- PB6 ($1004)	D6
8- PB7 ($1004)	D7
9- PC0 ($1003)	RS_AFF
10- PC1 ($1003)	E_AFF
11- PC2 ($1003)	
12- PC3 ($1003)	
13- PC4 ($1003)	
14- PC5 ($1003)	
15- PC6 ($1003)	
16- PC7 ($1003)	
17- +5V	
18- +5V	
19- 0V	
20- 0V	

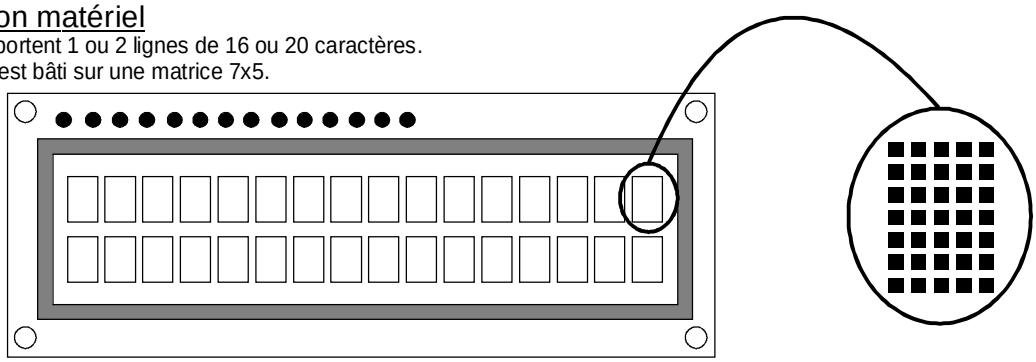
```
*****************************************************************
* declaration des peripheriques d'entree/sortie paralleles
*****************************************************************
porta  equ $1000
portb  equ $1004
portc  equ $1003
porte  equ $100a
ddrc  equ $1007

din_mem   equ $40    *************************
din_can    equ $20    * masques  *
memcan    equ $10    *   *
hdata      equ $08    * port A  *
dout       equ $02    *************************


cdepompe  equ   $80   *****************
selgam2   equ   $20    *    masques    *
selgam1   equ   $10    *                *
paramrap  equ   $04    *                *
e_aff      equ   $02    *    port C      *
rs_aff     equ   $01    ****************


t_clav     equ  $08         * masque port E *


*****************************************************************
***
*** SProgs d'initialisation: port C
***
*****************************************************************

init   ldaa     #%11111111
       staa     ddrc
       rts
```

L'afficheur est connecté aux ports B et C.
Quelle est la particularité du port C par rapport au port B?
Quelle est la configuration du port C dans le contrôleur?

Modules afficheurs LCD (Liquid Crystal Display)

1- Présentation matériel

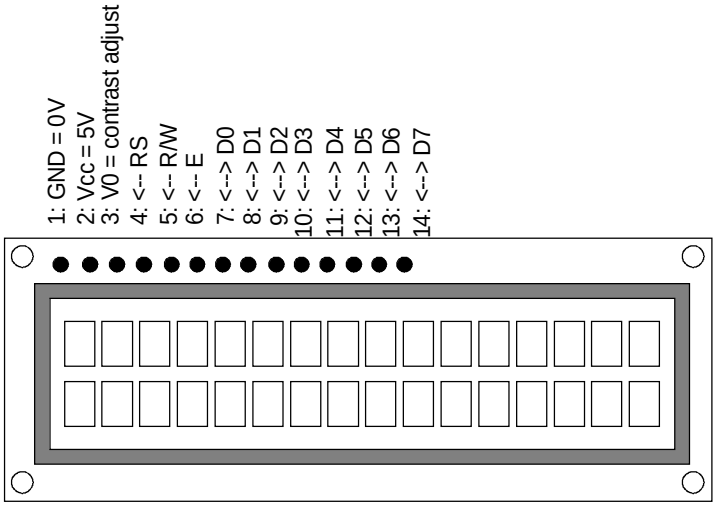
Ces modules comportent 1 ou 2 lignes de 16 ou 20 caractères.
Chaque caractère est bâti sur une matrice 7x5.



2- Liaisons électriques

Ils ont 14 broches
- 3 broches d'alimentation: Vss, Vdd et V0 (contrast adjust),
- 3 signaux de contrôle: RS (Register Select), R/W (Read/Write) et E (Enable),
- 8 broches pour le bus de données bidirectionnel,

(- optionnel: 2 broches pour le rétro-éclairage).



3- Principe de fonctionnement

Ces afficheurs sont "intelligents". Il est possible d'obtenir divers modes d'affichage.
Avant d'envoyer les caractères à afficher, il faut initialiser l'afficheur à l'aide d'***instructions***. Puis il est possible d'obtenir l'affichage de ***caractères***.
Les signaux RS et E permettent d'envoyer les instructions et les caractères.

En mode écriture avec R/W = 0:
- RS = 0 + impulsion au niveau haut de E (validation) permet l'envoi d'une instruction,
- RS = 1 + impulsion au niveau haut de E (validation) permet l'envoi d'un caractère,

Nota; l'impulsion au niveau haut de E doit avoir une durée minimum de 450ns.

En mode lecture avec R/W = 1: ce mode permet de venir lire le registre d'état de l'afficheur ou le contenu de la mémoire des caractères à afficher (DDRAM).

Les chronogrammes de la page 4/7 de la documentation montrent la chronologie des signaux RS et E. Le tableau de la page 5/7 montre les durées minimum des signaux à respecter. Pour éviter d'envoyer une instruction ou un caractère avant que le précédent ne soit complétement traité, on dispose de deux solutions:
- mettre le bus de données en lecture et tester le bit 7 du registre d'état (BF),
- utiliser des temporisations suffisantes. Il est alors possible de souder la broche R/W à la masse.

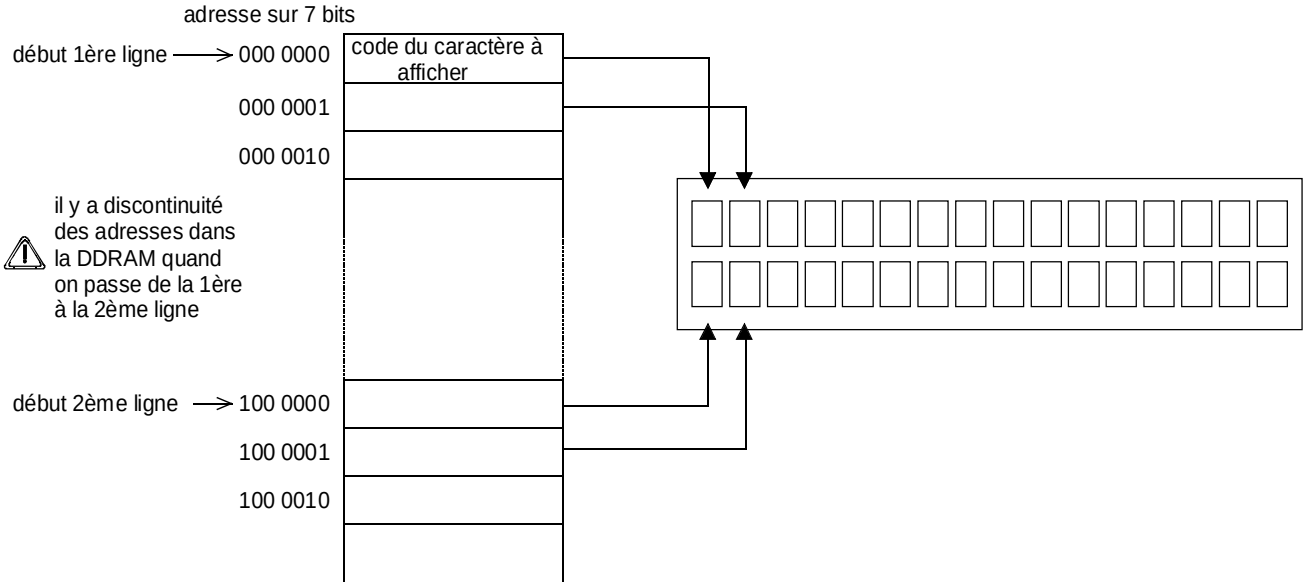
4. Etude des sous-programmes qui génèrent les signaux E et RS

```
*****************************************************************
* declaration des peripheriques d'entree/sortie paralleles
*****************************************************************
portb  equ $1004
portc  equ $1003
ddrc  equ $1007

cdepompe  equ   $80   *****************
selgam2   equ   $20   *   masques   *
selgam1   equ   $10   *             *
paramrap  equ   $04   *             *
e_aff     equ   $02   *   port C    *
rs_aff    equ   $01   ****************

*************************************************************
*** SProgs d'initialisation: port C
*************************************************************
init    ldaa     #%11111111
        staa     ddrc
        rts

* envoi & validation d'une instruction
* e: a  s: -  modif: -
*********************************************
instruc  staa    portb
         bclr    portc,y e_aff
         bclr    portc,y rs_aff
         bset    portc,y e_aff
         bclr    portc,y e_aff
         jsr     tp10ms
         rts
* envoi & validation d'un caractere
* e: a  s: -  modif: -
*********************************************
caract   staa    portb
         bclr    portc,y e_aff
         bclr    portc,y rs_aff
         bset    portc,y rs_aff
         bset    portc,y e_aff
         bclr    portc,y e_aff
         jsr     tp10ms
         rts

*** SProg de temporisation 10ms
*******************************************
tp10ms     pshx
           ldx    #3069
tp10ms1    dex
           bne    tp10ms1
           pulx
           rts

           END
```

- Etablir l'ordinogramme du sous-programme "instruc",
- Etablir l'ordinogramme du sous-programme "caract",
- Etablir l'ordinogramme du sous-programme "tp10ms",
- calculer la durée du sous-programme "10ms" sachant qu'un cycle machine machine dure 542,5 ns,
- tracer l'allure des signaux RS et E quand les sous-programmes "instruc" et "caract" sont exécutés,
- vérifier dans la documentation page 5/7 que les spécifications sur les durées minimum des signaux sont bien respectées.

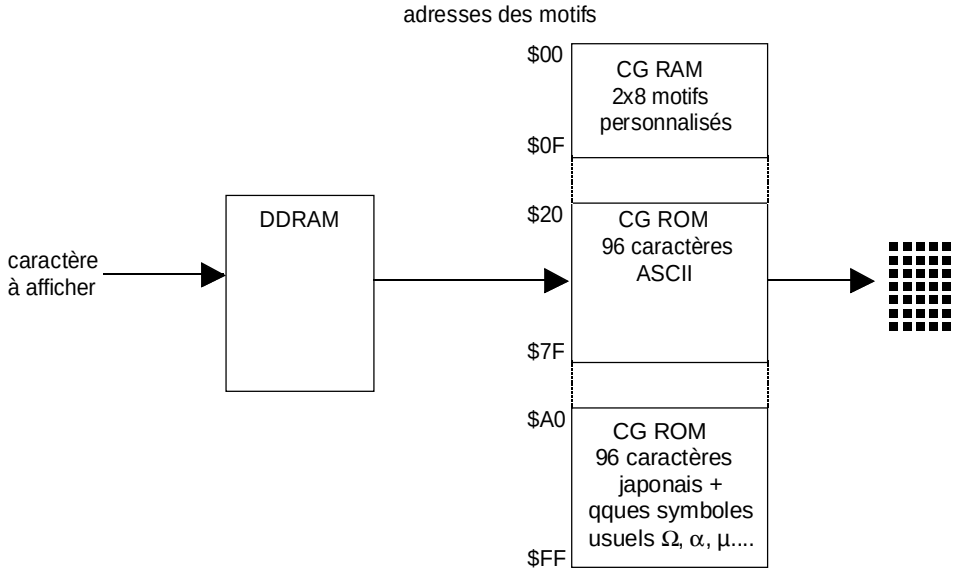
5- La DDRAM

C'est la mémoire intégrée à l'afficheur où l'on range les codes (ASCII ou autres) des caractères que l'on veut afficher.



6- CG ROM et CG RAM

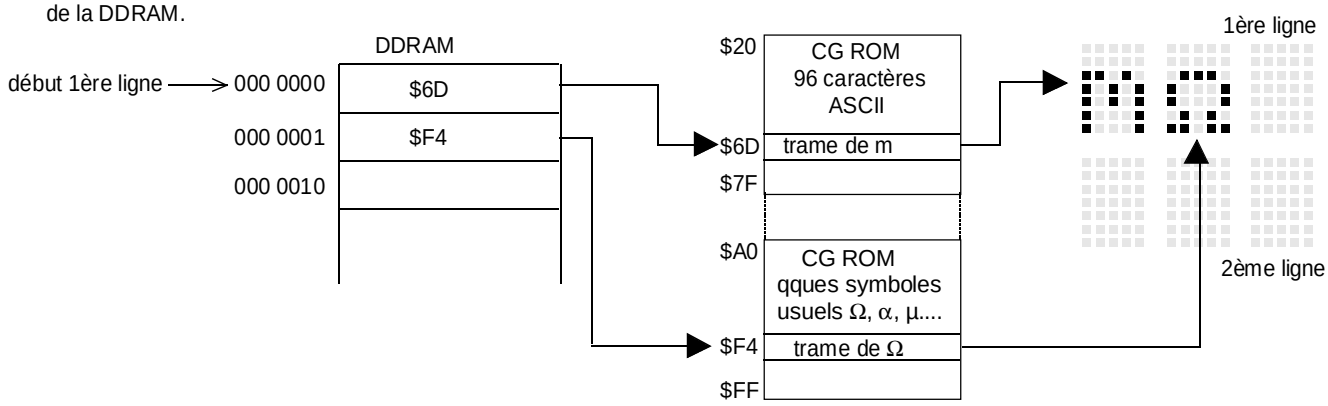
Les codes des caractères à afficher sont rangés dans la DDRAM (voir ci-dessus). Chaque code renvoit à un emplacement en CG ROM ou CG RAM (voir doc page3/7) contenant le motif de la matrice 7x5 à afficher.



Exemple

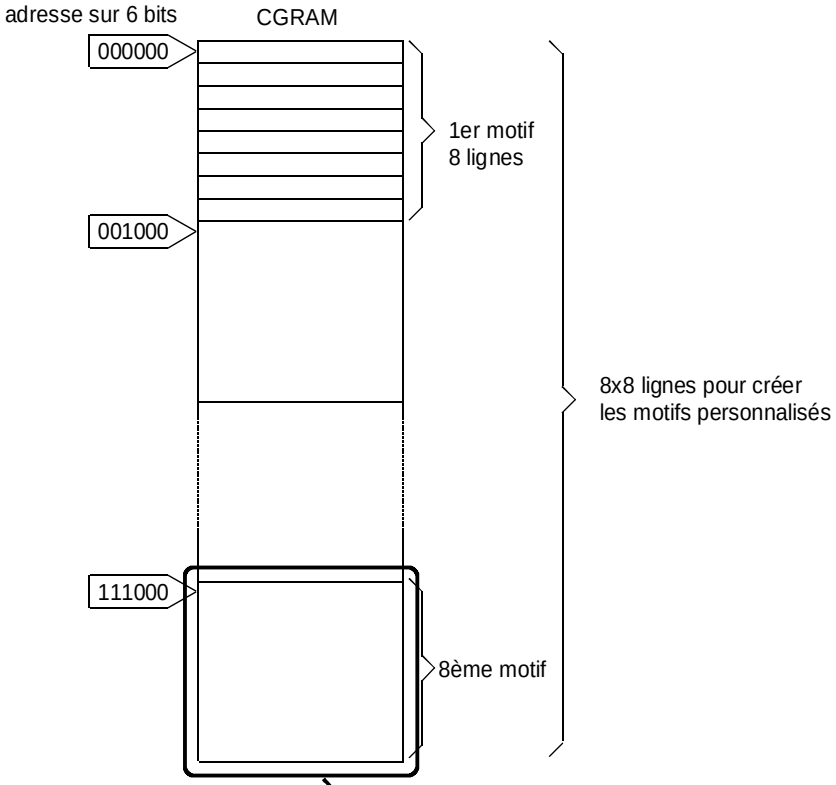
On veut afficher "mΩ" au début de la première ligne de l'afficheur:

- on envoit le code ASCII de "m" ($6D) à l'adresse %000 0000 de la DDRAM,
- on envoit le code $F4 (voir le générateur de caractères page 3/7 de la doc) correspondant au caractère "Ω" à l'adresse %000 0001 de la DDRAM.



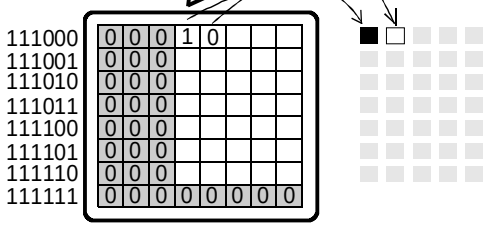
7- Caractères personnalisés

Il est possible de créer 8 motifs personnalisés. Ces motifs sont rangés dans la CGRAM sous forme de matrice 7x5.
Les adresses en CGRAM sont codées sur 6 bits



Le 1er motif correspond au code $00, il est répété au code $08,
le 2ème motif correspond au code $01, il est répété au code $09, etc....

Constitution d'un motif:



Procédure pour créer un motif personnalisé:
- il faut tout d'abord envoyer l'instruction permettant d'accéder à la CGRAM (instruction => RS = 0)

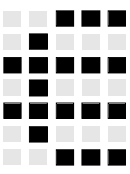
demande d'accès à la CGRAM:

D7	D6	D5	D4	D3	D2	D1	D0
0	1	---- adresse du motif----					

- ensuite on envoit les 8 lignes constituant le motif sous forme de caractères (caractère => RS = 1)

Exemple: on veut créer le symbole de l'Euro.

données								
$07	0	0	0	0	0	1	1	1
$08	0	0	0	0	1	0	0	0
$1F	0	0	0	1	1	1	1	1
$08	0	0	0	0	1	0	0	0
$1F	0	0	0	1	1	1	1	1
$08	0	0	0	0	1	0	0	0
$07	0	0	0	0	0	1	1	1
$00	0	0	0	0	0	0	0	0



8- Etude de l'initialisation de l'afficheur

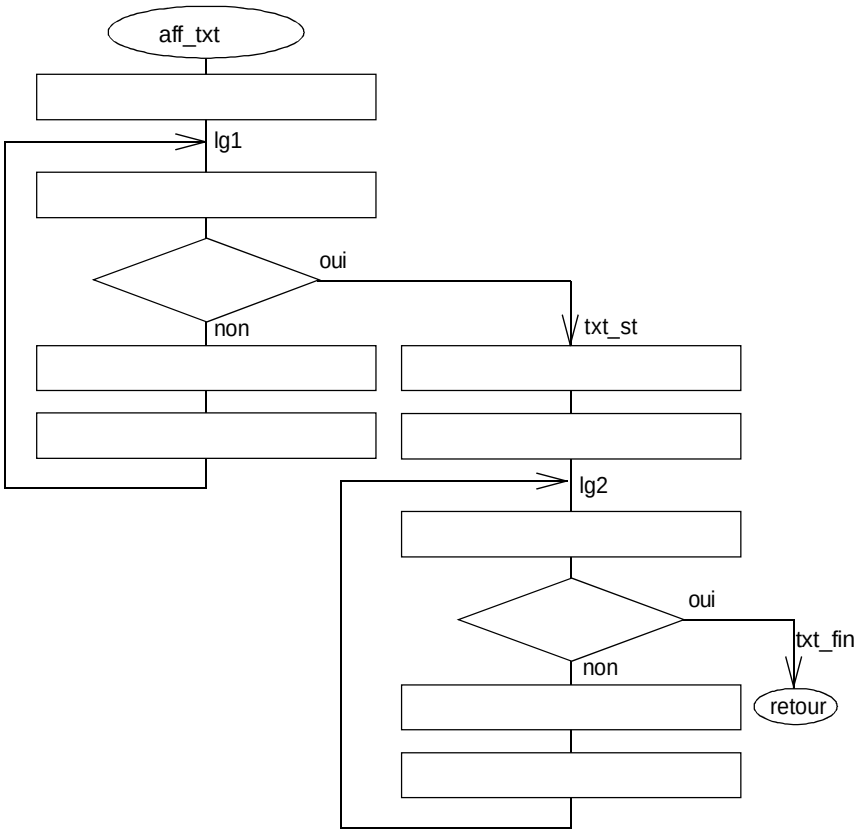
En vous aidant du tableau et des explications pages 5 et suivantes de la doc, retrouver la configuration de l'afficheur dans le programme suivant:

```
*********************************************
* initialisation de l'afficheur
* e: -  s: -  modif: a
*********************************************
init_af  ldaa     #%00111000
         jsr      instruc
         ldaa     #%00001100
         jsr      instruc
         ldaa     #%00000110
         jsr instruc
         rts
```

8- Etude du programme d'affichage d'un texte sur deux lignes

Faire l'analyse du programme principal ci-dessous et compléter l'ordinogramme du sous -programme aff_txt.

```
***********************************
*** programme principal
***********************************
        jsr init_af      ; initialisation de l'afficheur
        jsr clr_aff      ; effacement de l'afficheur
        ldx #texte       ; Pointeur du texte
        jsr aff_txt      ; afficher le texte
fin     bra fin


* envoi d'un texte sur deux lignes
* e: x  s: -  modif: a
*********************************************
aff_txt     ldaa     #%10000000
            jsr      instruc
lg1         ldaa      0,x
            beq      txt_st
            jsr      caract
            inx
            bra lg1
txt_st      ldaa     #%11000000
            jsr      instruc
            inx
lg2         ldaa      0,x
            beq      txt_fin
            jsr      caract
            inx
            bra      lg2
txt_fin     rts


* effacement complet de l'afficheur
* e: -  s: -  modif: a
*********************************************
clr_aff     ldaa     #%00000001
            jsr instruc
            rts

*   Texte à afficher *
* (table de caracteres) *
*************************
texte  fcc    "T"
       fcb    $00
       fcc    "STI"
       fcb    $00
```



En vous aidant du tableau des codes ASCII ci-dessous, afficher un texte sur deux lignes en utilisant la directive d'assemblage fcb.

Codes ASCII (American Standard Codes for Interchange Informations)

hexa	MSD	0	1	2	3	4	5	6	7
LSD	binary	000	001	010	011	100	101	110	111
0	0000	NUL	DLE	space	0	@	P	_	p
1	0001	SOH	DC1	!	1	A	Q	a	q
2	0010	STX	DC2	"	2	B	R	b	r
3	0011	ETX	DC3	#	3	C	S	c	s
4	0100	EOT	DC4	$	4	D	T	d	t
5	0101	ENQ	NAK	%	5	E	U	e	u
6	0110	ACK	SYN	&	6	F	V	f	v
7	0111	BEL	ETB	'	7	G	W	g	w
8	1000	BS	CAN	(	8	H	X	h	x
9	1001	HT	EM	)	9	I	Y	i	y
A	1010	LF	SUB	*	:	J	Z	j	z
B	1011	VT	ESC	+	;	K	[	k	{
C	1100	FF	FS	,	<	L	\	l	--
D	1101	CR	GS	-	=	M	]	m	}
E	1110	SO	RS	.	>	N	^	n	~
F	1111	SI	US	/	?	O	<-	o	DEL

MSD: Most Significant Digit (quartet de poids fort)
LSD: Least Significant Digit (quartet de poids faible)

Exemples: le code ASCII de 5 est 011 0101 (binaire) ou 35 (hexadécimal)
le code ASCII de M est 100 1101 (binaire) ou 4D (hexadécimal)
le code ASCII de l'espace est 010 0000 (binaire) ou 20 (hexadécimal)