AFFICHEUR LCD

Le module possède deux registres sélectionnés par RS que l'on peut lire ou écrire:

- REGISTRE D'ETAT ou D'INSTRUCTION (RS=0).
- REGISTRE DATA (RS=1).

◆ **LECTURE DU REGISTRE D'ETAT** : RS=0 et R/W=1

On sélecte le LCD par E=1 puis on lit les 8 bits du LCD. Le bit b7 de l'octet lu est le BUSY FLAG(BF). Si BF=1 cela signifie que le module LCD est occupé et ne peut pas recevoir de commande écriture. On doit attendre qu'il soit libre et le signale par BF=0. On peut alors lui écrire une instruction ou un caractère.

◆ **ECRITURE D'UNE INSTRUCTION :** RS = 0 et R/W = 0

On écrit les 8 bits de la DATA correspondant à l'instruction. On sélecte le LCD par E = 1 puis après un délai d'au moins 450 ns on désélecte le LCD par E = 0.

◆ **ECRITURE D'UN CARACTERE:** RS = 1 et R/W = 0

On écrit les 8 bits de la DATA correspondant au caractère. On sélecte le LCD par E = 1 puis après un délai d'au moins 450 ns on désélecte le LCD par E = 0

CODES INSTRUCTION:

◆ CLEAR DISPLAY = 0 0 0 0 0 0 0 1 = $01

◆ INIT FONCTION = 0 0 1 N L F 0 0

- F = fonte caractère: "0" = 5x7 "1" =5x11
- L = nombre de ligne : "0" = 1 ligne "1" = 2 lignes
- N = nombre de bit : "0" = 4bits "1" = 8 bits

◆ DISPLAY ON / OFF = 0 0 0 0 1 D C B

- B = Clignotement: "0" = non "1" =oui
- C = Curseur : "0" = OFF "1" = ON
- D = Display : "0" = OFF "1" = ON

◆ INIT MODE DISPLAY = 0 0 0 0 0 1 I S

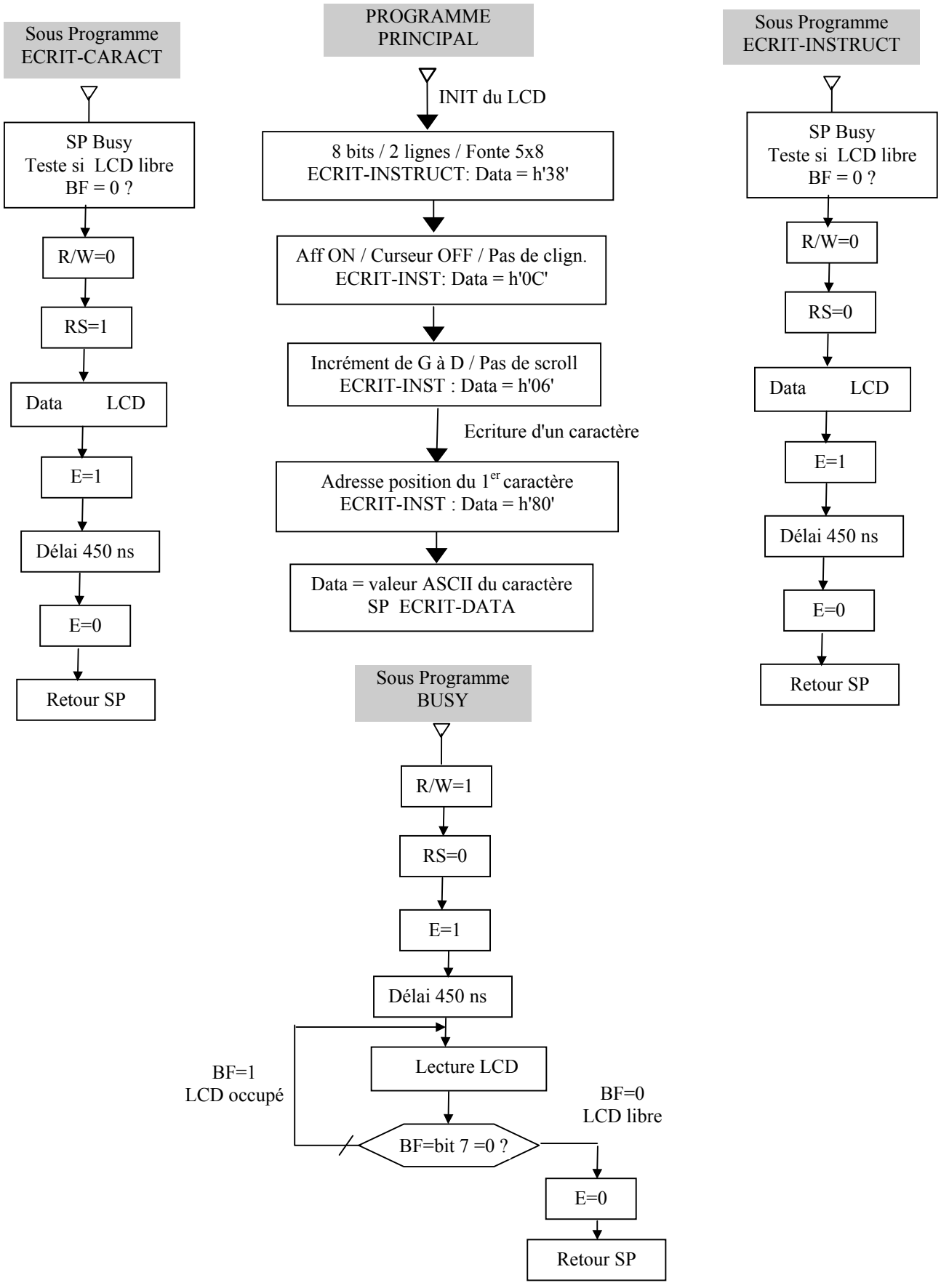
- S = Scrolling : "0" = display fixe "1" = scrolling display
- I = Incrément : "0" = de droite à gauche "1" = de gauche à droite

◆ ADRESSAGE de la DDRAM = 1 x x x x x x x Avec: xxxxxxx= n° de la case

1ere ligne :	case 0 $80	case 1 $81	.....	case 15 $8F	case 16 $90	.....	case 63 $BF
2eme ligne :	case 64 $C0	case 65 $C1	.....	case 79 $CF	case 80 $D0	.....	case 127 $FF

Seuls les 16 premiers caractères des lignes 1 et 2 sont affichés (case 80 à 8F et C0 à CF).

ORGANIGRAMME



AFFICHEUR 4 Lignes 20 Caractères

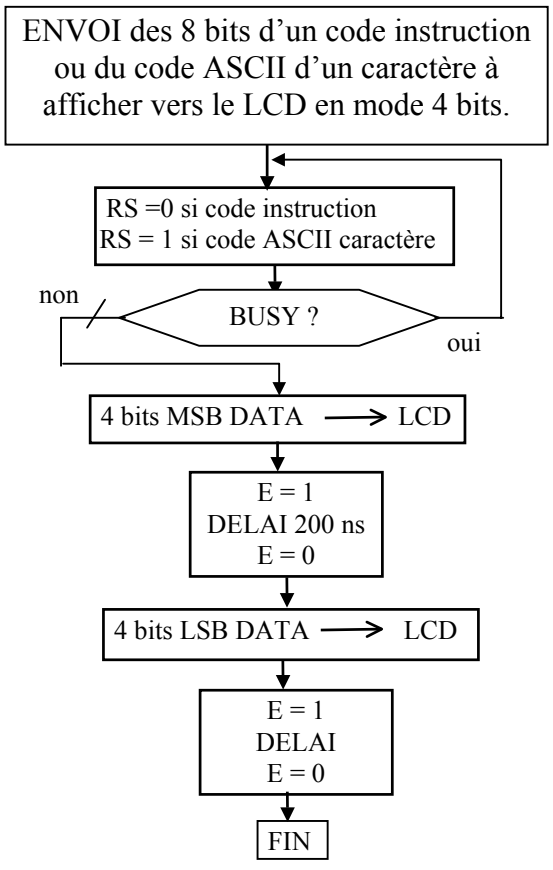
1ere ligne	case 0 $80	case 1 $81	.....	case 18 $92	case 19 $93
2eme ligne	case 20 $C0	case 21 $C1	.....	case 38 $D2	case 39 $D3
3eme ligne	case 40 $94	case 41 $95	.....	case 58 $A6	case 59 $A7
4eme ligne	case 60 $D4	case 61 $D5	.....	case 78 $E6	case 79 $E7

GESTION DU LCD en 4 BITS

Les données sont envoyées en deux fois 4 bits. Le MSB en tête et le LSB ensuite. L'entrée des data se fait sur : D_4 D_5 D_6 et D_7 .
Après la mise sous tension un reset interne se fait dans le LCD. Il faut attendre 30 ms avant d'envoyer des DATA.
On envoie ensuite sur 4 bits : $3 (mode 8bits) 3 fois de suite avec un délai entre de 15 ms. Puis on envoi $ 2 (mode 4 bits) et ensuite le code INIT = $ 28 en 2 fois 4 bits soit : $ 2 puis $ 8.

INIT du LCD en 4 bits

- Reset par mise sous tension
- Attente min de 30 ms
- $3 ⟶ LCD (mode 8 bits)
- Attente 15 ms
- $3 ⟶ LCD (mode 8 bits)
- Attente 15 ms
- $3 ⟶ LCD (mode 8 bits)
- Attente 15 ms
- $2 ⟶ LCD (mode 4 bits)
- $2 ⟶ LCD
- $8 ⟶ LCD } $28 = Init Fonction 4 bits / 2 lignes
- $0 ⟶ LCD
- $C ⟶ LCD } $0C = Display On / Cursor off
- $0 ⟶ LCD
- $6 ⟶ LCD } $06 = Mode Incr. de G vers D
- $0 ⟶ LCD
- $1 ⟶ LCD } Clear écran LCD
- MSB Position caractère ⟶ LCD
- LSB Position caractère ⟶ LCD
- MSB Caractère ⟶ LCD
- LSB Caractère ⟶ LCD



AFFICHAGE D'UNE CHAINE DE CARACTERES SUR UN LCD

Utilise 2 variables à définir au préalable en RAM: PTR et CAR

```
;****************************************
;**     UTILISATION de la DIRECTIVE  "DT"  **
;**   qui crée des RETLW avec la valeur ASCII   **
;**         du caractère à afficher dans W            **
;****************************************
;
TABLE       ADDWF       PCL                 ; ajoute W au compteur ordinal donc saut de W pas
CHAINE1     DT          "ENAC",0FF          ; crée des RETLW avec la valeur ASCII
CHAINE2     DT          "sub ELE",0FF       ; la fin de chaîne est signalée par h'FF'

;************************************
;**          PROGRAMME  PRINCIPAL       **
;************************************
;
PROG  CALL        LCD84                     ; SP qui initialise LCD en mode 8 bits et 4 lignes
      MOVLW       086                       ; positionne écriture en 1ere ligne, 7eme position.
      CALL        ECRINS                    ; SP qui envoi W dans le registre instruction du LCD
      MOVLW       CHAINE1-TABLE-1           ; W contient offset entre début de table et CHAINE1
      CALL        AFTAB                     ; affiche chaine1 sur LCD
      MOVLW       0C5                       ; positionne écriture en 2eme ligne, 6eme position.
      CALL        ECRINS                    ; SP qui envoi W dans le registre instruction du LCD
      MOVLW       CHAINE2-TABLE-1           ; W contient offset entre début de table et CHAINE2
      CALL        AFTAB                     ; affiche chaine2 sur LCD
ATT   GOTO        ATT

;*******************************************
;**                  SP   AFTAB                  **
;**          Affiche les caractéres ASCII        **
;**        d'une chaine dont l'offset par rapport    **
;**              au début TABLE est dans W.          **
;**                 Fin de la table par h'FF'        **
;*******************************************
;
AFTAB       MOVWF       PTR                 ; offset début de la table dans PTR
AFTAB1      MOVFW       PTR                 ; restaure le pointeur de table dans W
            CALL        TABLE               ; récupére dans W le caractére ASCII à afficher
            MOVWF       CAR                 ; sauve le caractère
            SUBLW       h'FF                ; caractére = h'FF'? = fin de table ?
            BTFSC       Z                   ; si oui Z=1 si non Z=0
            GOTO        AFTAB2              ; oui car Z=1 donc c'est fini
            MOVFW       CAR                 ; non car Z=0 donc on restaure le caractère dans W
            CALL        ECRCAR              ; SP qui envoi W dans le registre caratère du LCD
            INCF        PTR                 ; incrémente le pointeur de table.
            GOTO        AFTAB1              ; on continue à écrire toute la table.
AFTAB2      RETURN
```