

# *SERIElectronique*

---

## SEMLAB1-B

# SOMMAIRE

<b>INTRODUCTION</b> .....	<b>5</b>
<b>LE CONNECTEUR VERS LA CARTE MERE</b> .....	<b>6</b>
<b>PRESENTATION DES BORNIERES A VIS</b> .....	<b>7</b>
<b>REGLAGE DES POTENTIOMETRES ET DES SWITCHS</b> .....	<b>8</b>
<b>PLAN DE L' ESPACE MEMOIRE</b> .....	<b>9</b>
<b>I - CLAVIER TELEPHONE</b> .....	<b>11</b>
1. LIAISON MICROCONTROLEUR VERS LE CLAVIER TELEPHONE .....	11
2. DESCRIPTIF DU FONCTIONNEMENT .....	11
3. DESCRIPTIF DU PROGRAMME P_TESTCLAV.A51 .....	12
3.1 Organigramme du programme principal (Figure 1) .....	12
3.2 Sous programme TEST_CLAV (S_CLAVIER.A51).....	12
3.3 Sous programme TEST_TOUCH.....	14
3.4 Sous programme AFFI_CAR (S_afilcd.A51) .....	15
3.5 Sous programme DEFI_LED (S_sprled.a51).....	15
<b>II - AFFICHEUR LCD</b> .....	<b>16</b>
1. LIAISON MICROCONTROLEUR VERS L'AFFICHEUR LCD .....	16
2. DESCRIPTIF DU FONCTIONNEMENT .....	16
3. DESCRIPTION DU FICHIER A INCLURE S_AFILCD.A51.....	16
3.1 Sous programme INIT_LCD (Figure 1) .....	17
3.2 Sous programme CURS_DEB (Figure 2).....	18
3.3 Sous programme CURS_DEB_L2 (Figure 3).....	18
3.4 Sous programme CURS_OFF (Figure 4).....	18
3.5 Sous programme SEND_CMDE (Figure 5).....	19
3.6 Sous programme AFFI_CAR (Figure 6).....	19
3.7 Sous programme WAI_BUZY (Figure 7).....	19
<b>III - SORTIES RELAIS (REL1 ET REL2)</b> .....	<b>20</b>
1. LIAISON MICROCONTROLEUR VERS LES SORTIES RELAIS .....	20
2. DESCRIPTIF DU FONCTIONNEMENT .....	20
3. DESCRIPTION DU PROGRAMME P_RELAIS.A51.....	21
3.1 Organigramme du programme principal (figure 1).....	21
<b>IV - ENTREES OPTOCOUPLEES (TIL199)</b> .....	<b>22</b>
1. LIAISON MICROCONTROLEUR VERS LES ENTREES OPTOCOUPLEES .....	22
2. DESCRIPTIF DU FONCTIONNEMENT .....	22
3. DESCRIPTION DU PROGRAMME P_RELAISOPTO.A51 .....	23
3.1 Organigramme du programme ( Figure 1 ).....	24
<b>V - CONVERTISSEURS AD ET DA RAPIDES AD7569 (Z22)</b> .....	<b>25</b>
1. LIAISON MICROCONTROLEUR VERS LE CONVERTISSEUR AD/DA.....	25
2. DESCRIPTIF DU FONCTIONNEMENT .....	25
3. DESCRIPTION DU PROGRAMME P_CONVAD-DA.A51 .....	26
3.1 Organigramme du programme principal ( Figure 1 ).....	26
3.2 Sous programmes INIT_LCD, CURS_DEB, CURS_OFF et AFFI_CAR.....	27
3.3 Sous programme FORME (Figure 2) .....	28
3.4 Sous programme AFF_TEXTE (Figure 3).....	30
<b>VI - HORLOGE TEMPS REEL (DS-12887)</b> .....	<b>31</b>
1. LIAISON MICROCONTROLEUR VERS LE CIRCUIT RTC - HORLOGE TEMPS REEL .....	31
2. DESCRIPTIF DU FONCTIONNEMENT .....	31
3. DESCRIPTIF DU PROGRAMME P_INIT_HTR.A51 .....	32
3.1 Organigramme du programme P_INIT_HTR.A51 (figure 1).....	33
4. DESCRIPTIF DU PROGRAMME P_HTR_IT.A51 .....	33

4.1	Configuration du circuit RTC.....	33
4.2	Déroulement du programme.....	34
4.3	Organigramme du sous programme CONFIG (figure 2).....	36
4.4	Organigramme du sous programme F_date (figure 3).....	36
4.5	Organigramme du sous programme F_heure (figure 4).....	37
4.6	Organigramme du sous programme S_digit (figure 5).....	38
<b>VII - PANNEAU DE LED.....</b>		<b>39</b>
1.	LIAISON MICROCONTROLEUR VERS LE PANNEAU DE LEDS.....	39
2.	DESCRIPTIF DU FONCTIONNEMENT.....	39
3.	DESCRIPTIF DU PROGRAMME P_MESLED.A51.....	41
3.1	Organigramme du programme principal (Figure 1).....	41
3.2	Sous programme GET_MSGE (Figure 2).....	41
3.3	Sous programme AFFIMES (Figure 3).....	43
4.	DESCRIPTIF DU FICHIER A INCLURE S_ASCII.A51.....	43
4.1	Sous programme CONV_LED (Figure 5).....	43
4.2	Sous programme WRITE_STRING (figure 5).....	44
5.	DESCRIPTIF DU FICHIER A INCLURE S_AFILED.A51.....	45
5.1	Sous programme INIT_LED (Figure 5).....	45
5.2	Sous programme AFI_ECRAN (Figure 6).....	46
5.3	Sous programme AFI_LIGNE (figure 7).....	47
6.	DESCRIPTIF DU FICHIER A INCLURE S_SPRLED.A51.....	48
6.1	Sous programme DECAL_MEM (figure 8).....	48
6.2	Sous programme DEFI_LED (figure 9).....	49
<b>VIII - COMMUNICATION IRDA SCC2691(Z15), HSDL7001(Z23) ET HSDL1001(Z24).....</b>		<b>50</b>
1.	LIAISON MICROCONTROLEUR VERS L'INTERFACE IRDA.....	50
2.	DESCRIPTIF DU FONCTIONNEMENT.....	50
3.	DESCRIPTIF DU PROGRAMME P_IRDACHIF.A51.....	51
3.1	Organigramme du programme P_IrDACHif.A51 (Figure 1).....	51
3.2	Sous programme INIT_SCI (Figure 3).....	52
3.3	Sous programme CLRFIFO (Figure 4).....	53
3.4	Sous programme EMISS_IRDA (Figure 4).....	53
3.5	Sous programme RECP_IRDA (Figure 5).....	54
3.6	Sous programme CLEAR_STAT_ERR (figure 6).....	54
3.7	Sous programme AFFICH (P_IrDACHif.A51) (figure 7).....	55
4.	DESCRIPTIF DU PROGRAMME P_IRDA_CHAR_IT.A51.....	55
4.1	Organigramme du programme P_IrDAChar_IT.A51 (Figure 8).....	56
4.2	Sous programme INIT_SCI (figure 9).....	57
4.3	Sous programme INTERRUPT (figure 10).....	57
4.4	Sous programme RECEP (figure 11).....	58
4.5	Sous programme EMISS_IRDA.....	59
4.6	Sous programme WAIT_RECEP (figure 12).....	59
4.7	Sous programme AFFICH (P_IrDAChar_IT.A51) (figure 13).....	60
<b>IX - CONTROLEUR DE BUS I<sup>2</sup>C (PCF8584).....</b>		<b>61</b>
1.	LIAISON MICROCONTROLEUR VERS LE CONTROLEUR DE BUS I <sup>2</sup> C.....	61
2.	DESCRIPTIF DU CIRCUIT PCF8584.....	61
3.	DESCRIPTIF DE LA PROGRAMMATION DU PCF8584.....	67
<b>IX A PORT D'ENTREES-SORTIES 8 BITS PROGRAMMABLE ( PCF8574 ).....</b>		<b>71</b>
1.	LIAISON MICROCONTROLEUR VERS LE PORT D'ENTREES/SORTIES 8 BITS PROGRAMMABLE.....	71
2.	DESCRIPTIF DU FONCTIONNEMENT.....	71
3.	DESCRIPTIF DU PROGRAMME P_I2C_E-S.A51.....	72
3.1	Organigramme (Figure 1).....	72
3.2	Sous programme TEST_CLAV.....	72
3.3	Sous programme TEST_TOUCH.....	74
3.4	Sous programme CHOIX_PRGM.....	74
<b>IX B - CONVERTISSEUR AD / DA ( PCF8591 ).....</b>		<b>76</b>
1.	LIAISON MICROCONTROLEUR VERS LE CONVERTISSEUR AD / DA.....	76
2.	DESCRIPTIF DU FONCTIONNEMENT.....	76
3.	DESCRIPTIF DU PROGRAMME P_ANALOG.A51.....	77
<b>IX C - CAPTEUR DE TEMPERATURE ( DS1621 ).....</b>		<b>78</b>

1.	LIAISON MICROCONTROLEUR VERS LE CAPTEUR DE TEMPERATURE .....	78
2.	DESCRIPTIF DU FONCTIONNEMENT .....	78
3.	DESCRIPTIF DU PROGRAMME P_CAPTEMP.A51.....	79
3.1	<i>Organigramme du programme principal (Figure 1 )</i> .....	79
3.2	<i>Configuration du DS1621</i> .....	79
3.3	<i>Configuration du thermostat (sous programme THERMOSTAT )</i> .....	81
3.4	<i>Attente de fin de conversion (sous programme CONV_DONE )</i> .....	81
3.5	<i>Lecture de la température</i> .....	81
3.6	<i>Mise en forme de la température (sous programme FORME )</i> .....	82

# INTRODUCTION

La carte référencée ‘SEMLAB1-B’ est une extension spécifique aux microsystemes MC711, MC12, MC51TT et MC51. Elle se connecte à ces systemes à travers le connecteur CONN\_SEMLAB.

Cette extension est équipée de :

- 1 clavier à 12 touches
- 1 afficheur LCD de 2 lignes, 20 caractères par ligne
- 1 panneau d'affichage de 128 leds organisées en 8 lignes de 16 colonnes
- 2 convertisseurs (un AD et un DA) rapides
- 1 circuit I<sup>2</sup>C avec 1 sortie et 4 entrées analogiques
- 1 circuit I<sup>2</sup>C avec 8 entrées-sorties TTL programmables
- 1 circuit d'interface I<sup>2</sup>C parallèle
- 1 capteur de température sur bus I<sup>2</sup>C
- 2 sorties relais
- 4 entrées optocouplées
- 1 circuit de communication IrDA (Récepteur et émetteur)
- 1 Horloge temps réel sur la carte MC51TT

# LE CONNECTEUR VERS LA CARTE MERE

La carte MC51-TT dispose d'un connecteur d'extension de 40 points nommé CONN-SEMLAB afin de connecter la carte d'extension SEMLAB1-B.

Vu de la carte SEMLAB1-B, on dispose sur ce connecteur du bus de données, du bus d'adresses et des signaux de contrôle du microcontrôleur.

## BROCHAGE DU CONNECTEUR

<b>+12 Volts</b>	1	40	<b>Masse</b>
(1) /RESET	2	39	- nc -
- nc -	3	38	/RD
/WR	4	37	- nc -
- nc -	5	36	- nc -
- nc -	6	35	/CS_SEMLAB (2)
- nc -	7	34	- nc -
- nc -	8	33	- nc -
- nc -	9	32	- nc -
- nc -	10	31	- nc -
ADR_07	11	30	ADR_06
ADR_05	12	29	ADR_04
ADR_03	13	28	ADR_02
ADR_01	14	27	ADR_00
DATA_7	15	26	DATA_6
DATA_5	16	25	DATA_4
DATA_3	17	24	DATA_2
DATA_1	18	23	DATA_0
<b>IRQ</b>	19	22	<b>XRQ</b>
<b>Masse</b>	20	21	<b>+5 Volts</b>

(1) : /RESET (actif au niveau bas) est la sortie /RSTOL du microcontrôleur 80C390

(2) : CS\_SEMLAB est un « chip-select » qui décode dans l'espace DATA aux adresses [8000 ... 80FF].

## PRESENTATION DES BORNIERES A VIS

SCL	}	Entrée Serial Clock pour le bus I <sup>2</sup> C
SDA	}	Entrée Sériale DATA
M	}	GND
OUT	}	Sortie analogique du convertisseur AN/NA PCF8591
E0	}	Entrées multiplexées analogiques du convertisseur AN/NA PCF8591
E1		
E2		
E3		
REL1	}	Circuit Relais 1
REL1		
REL2	}	Circuit Relais 2
REL2		
I4 +	}	Entrées optocouplées
I4 -		
I3 +		
I3 -		
I2 +		
I2 -		
I1 +		
I1 -		
VDA	}	Sortie analogique du convertisseur AD7569
VAD	}	Entrée analogique du convertisseur AD7569

## REGLAGE DES POTENTIOMETRES ET DES SWITCHS

P1 et P2 sont des potentiomètres situés en dessous du bornier et à droite de l'afficheur LCD, ils permettent de régler le circuit intégré AD7569 (Voir le chapitre concernant le convertisseur AD7569)

L'interrupteur SW2 situé juste en dessous des deux potentiomètres est utilisé pour configurer le port infrarouge.

Le potentiomètre P3 situé à droite de la touche 3 du clavier 12 touches permet de régler le niveau du contraste de l'afficheur LCD

L'interrupteur SW1 situé en plein milieu de la carte permet de choisir le mode de fonctionnement de la led LED1. Si SW1 est réglé en position 1 , la LED est utilisée par le capteur de température DS1621, Si SW1 est réglé en position 2 , la LED est utilisée pour l'affichage de l'état des relais.

## PLAN DE L' ESPACE MEMOIRE

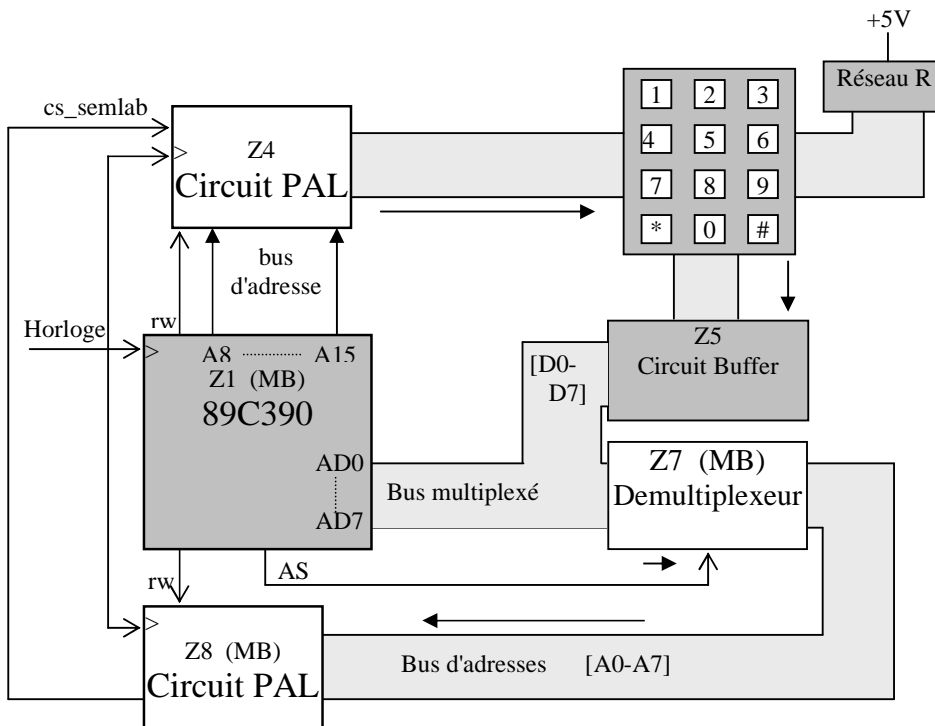
Chaque composant possède une ou plusieurs adresses dans l'espace XDATA pour être sélectionnée :

ENTITE ADRESSEE DES PERIPHERIQUES	ADRESSE LOCALE
Convertisseur AD/DA AD7569	8000
Bus I <sup>2</sup> C (PCF8584)	8002
Registre S1 du circuit I <sup>2</sup> C PCF8584	8003
Colonnes du panneau de leds	8004
Lignes du panneau de leds	8008
Lignes du clavier	800A
Colonnes du clavier	800B
Instruction de l'afficheur LCD	800C
Données de l'afficheur LCD	800D
UART (Port Infrarouge)	8010
Relais 1 (Ecriture) Entrée optocouplées 1 et 2 (Lecture)	8018
Relais 2 (Ecriture) Entrée optocouplées 3 et 4 (Lecture)	8019
HSDL-7001 UART (Port Infrarouge)	801A



# I - Clavier téléphone

## 1. Liaison microcontrôleur vers le clavier téléphone



(MB) : Composants se trouvant sur la carte mère MC51TT  
 Les autres composants se trouvent sur la carte SemLab1-B

## 2. Descriptif du fonctionnement

Le clavier comporte 7 fils reliés à la fois au bus de données du microcontrôleur (Tableau 1) par l'intermédiaire du buffer Z5 mais aussi au PAL Z4.

Les bits b2, b1, b0 représentent les colonnes de gauche à droite.

Les bits b6, b5, b4, b3 représentent les lignes de bas en haut.

Colonnes :	3	2	1	Lignes	Bus de données
	1	2	3	4	b6
	4	5	6	3	b5
	7	8	9	2	b4
	*	0	#	1	b3
Bus de données	b2	b1	b0		

**Tableau 1** : Correspondance fils reliés au clavier et bus de donnés

Bus de données	D7	D6	D5	D4	D3	D2	D1	D0
Clavier	X	L4	L3	L2	L1	C3	C2	C1

Pour déterminer si une touche du clavier est appuyée ainsi que ses coordonnées, il faut mettre les colonnes à 0 et lire les lignes. Si une des lignes est à 0, il y a une touche enfoncée ; on met alors les lignes à 0 et on lit les colonnes. Il ne doit y avoir qu'un seul bit à 0 à chaque lecture (ligne puis colonne). Ce principe est assuré par la PAL Z4. La lecture des lignes se fait à l'adresse 800Ah et la lecture des colonnes se fait à l'adresse 800Bh.

Voir le fichier P\_TestClav.a51 et S\_CLAVIER.a51.

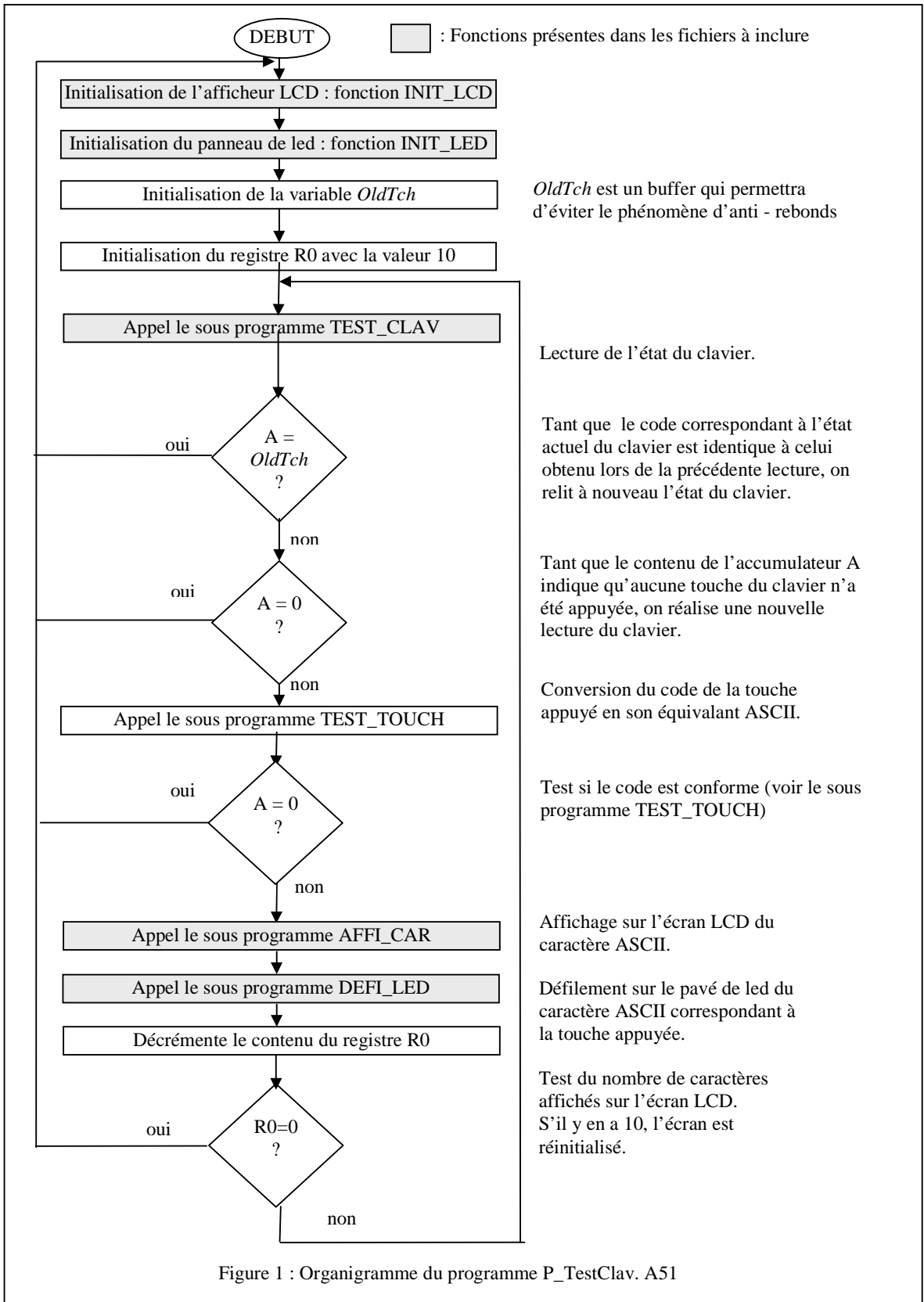
### 3. Descriptif du programme P\_TestClav.A51

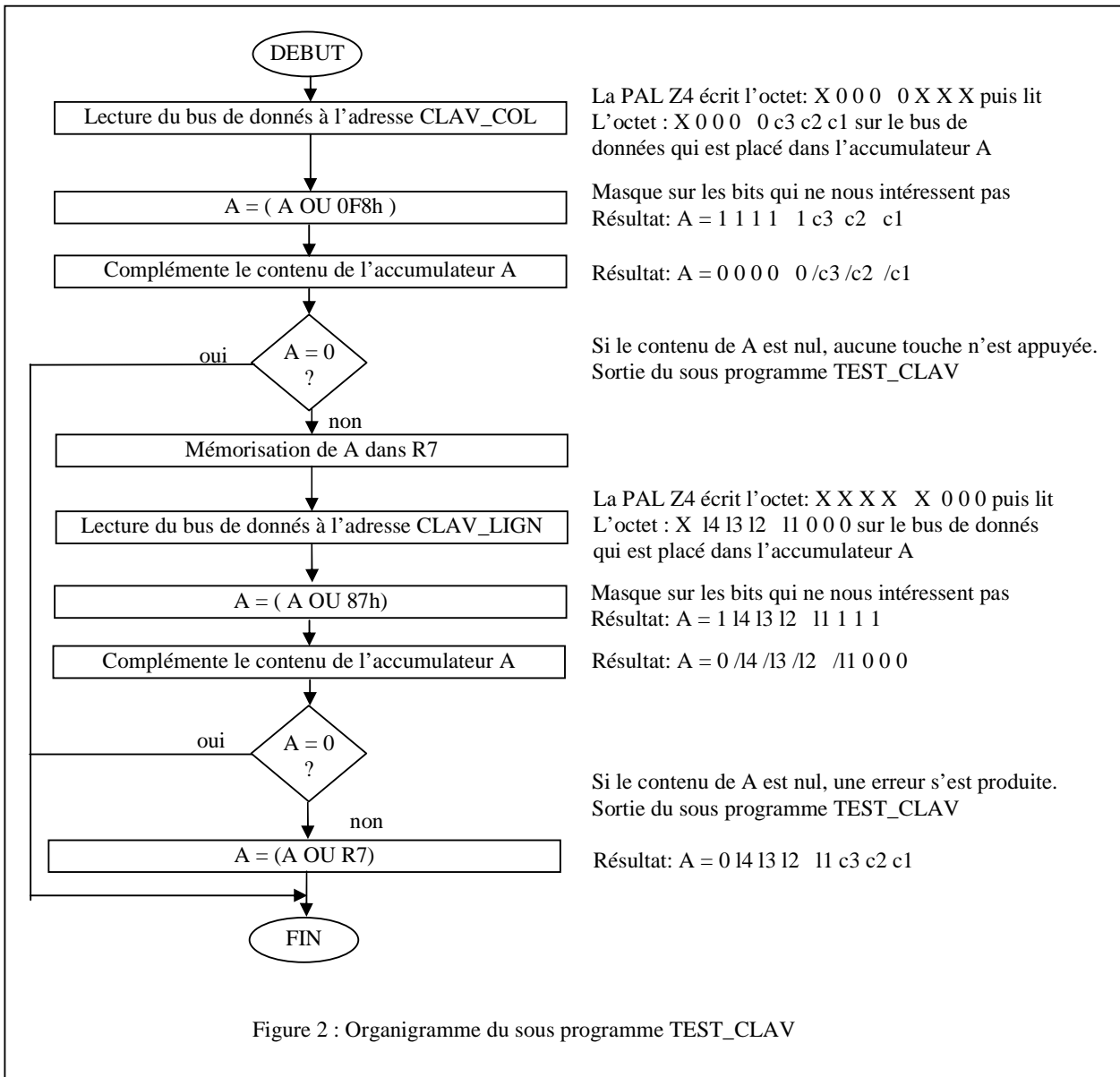
Ce programme permet d'afficher les touches du clavier sur l'afficheur LCD mais aussi sur le panneau de leds. L'afficheur LCD permet l'affichage de toutes les touches du clavier dans la limite de 10 avant d'être effacé, ce qui est également réalisé après l'appui sur l'une des deux touches '#' ou '\*'. Le panneau de leds permet l'affichage et le défilement des touches (uniquement numériques) une par une.

#### 3.1 Organigramme du programme principal (Figure 1)

#### 3.2 Sous programme TEST\_CLAV (S\_CLAVIER.A51)

Cette fonction permet de lire l'état actuel du clavier et de fournir un code correspondant à l'appui sur chacune des touches ainsi que pour le cas où aucune touche ne serait appuyée comme le montre le tableau 2. Cette conversion de l'état du clavier en un code s'effectue en plusieurs étapes (Figure 2).





### 3.3 Sous programme TEST TOUCH

Le sous programme TEST\_TOUCH permet à partir du code fourni par la fonction TEST\_CLAV de le convertir en son code ASCII qui est placé dans l'accumulateur A en vue de son affichage sur l'écran LCD par la fonction AFFI\_CAR ( S\_AFILCD.A51 ). Il utilise également le pointeur DPTR pour pointer sur les chaînes de caractères qui contiennent les octets nécessaires à l'affichage et au défilement des touches numériques du clavier par l'intermédiaire de la fonction DEFI\_LED ( S\_SPRLED.A51 ). Chacune des chaînes de caractères y compris la chaîne de caractères vide *novalsl* se termine par le code hexadécimal 0BBh qui sera utile pour connaître la fin de la chaîne. Par conséquent la taille d'un caractère à afficher sur le pavé de LED est extensible

Tableau 2 : Sous programme TEST\_TOUCH

CODES DU CLAVIER (HEXADECIMAL)	CODE ASCII CORRESPONDANT (HEXADECIMAL)	CHAINE DE CARACTERE POINTEE PAR DPTR (S_SPRLED.A51)
00h	30h	<i>val0sl</i>
44h	31h	<i>val1sl</i>
42h	32h	<i>val2sl</i>
41h	33h	<i>val3sl</i>
24h	34h	<i>val4sl</i>
22h	35h	<i>val5sl</i>
21h	36h	<i>val6sl</i>
14h	37h	<i>val7sl</i>
12h	38h	<i>val8sl</i>
11h	39h	<i>val9sl</i>
0Ch	2Ah	<i>novalsl</i>
0Ah	30h	<i>val0sl</i>
09h	23h	<i>noval</i>

### 3.4 Sous programme AFFI\_CAR (S\_afilcd.A51 )

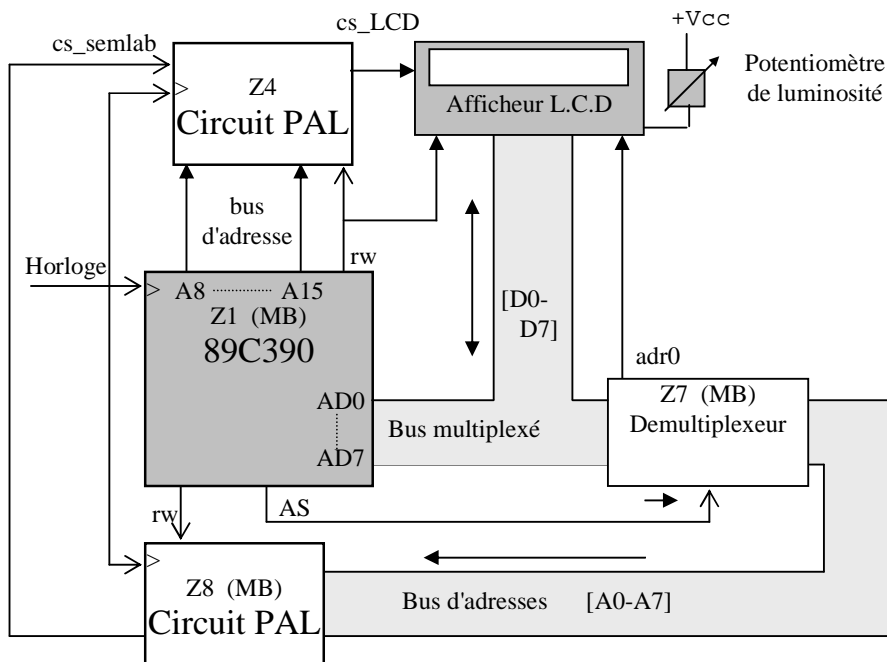
Ce sous programme permet l'affichage des caractères ASCII fournis par la fonction TEST\_TOUCH qui se trouvent dans l'accumulateur A sur l'afficheur LCD. Son fonctionnement est expliqué dans le chapitre concernant l'afficheur LCD.

### 3.5 Sous programme DEFI\_LED (S\_sprled.a51)

Le sous programme DEFI\_LED permet l'affichage et le défilement de tout caractère numérique ou d'une chaîne de caractères sur le panneau de leds. Son fonctionnement est expliqué dans le chapitre concernant le pavé de led.

## II - Afficheur LCD

### 1. Liaison microcontrôleur vers l'afficheur LCD



(MB) : Composants se trouvant sur la carte mère MC 51TT  
Les autres composants se trouvent sur la carte SemLab1-B

### 2. Descriptif du fonctionnement

L'écriture sur l'afficheur LCD d'un caractère ASCII à la position courante du curseur se fait par une écriture à l'adresse 800Dh. Pour envoyer une commande à l'afficheur (par exemple déplacement du curseur, effacer l'écran, activer ou désactiver le curseur,...), on écrit le code de la commande à l'adresse 800Ch, ce qui transfère l'ordre à l'afficheur.

### 3. Description du fichier à inclure S\_AFILCD.A51

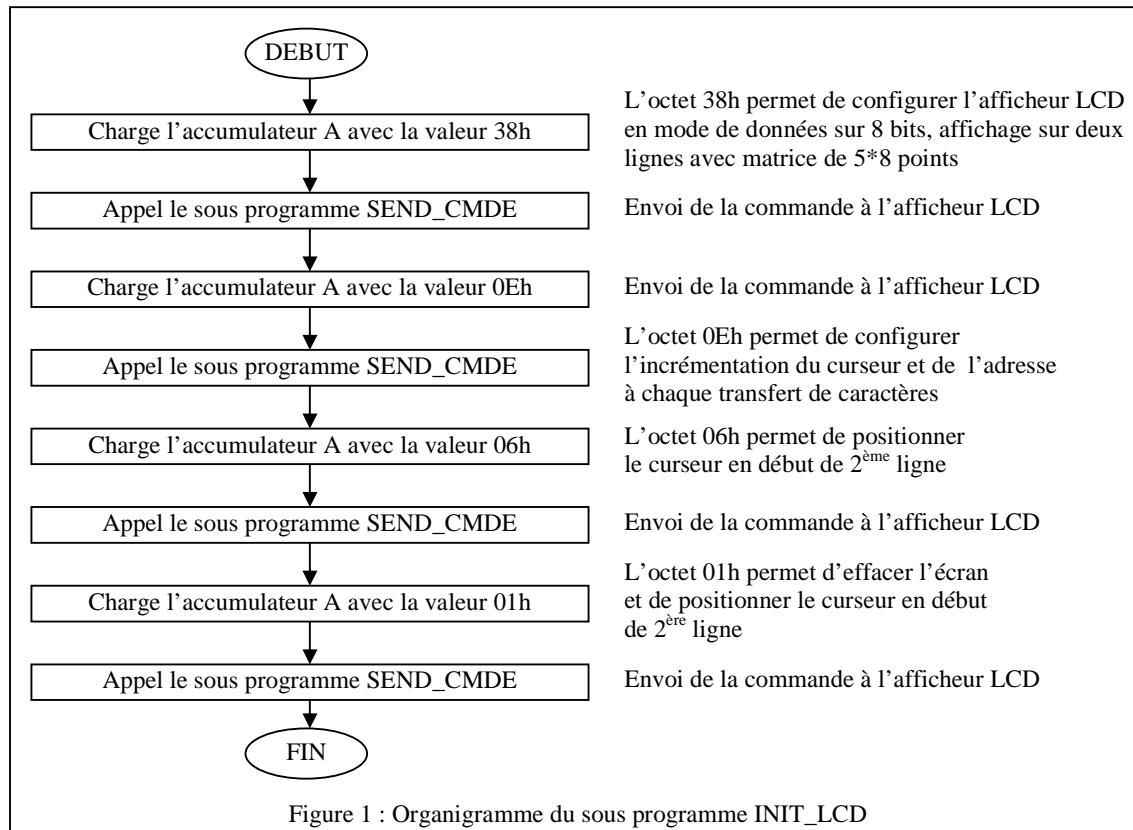
Le fichier S\_AFILCD.A51 est constitué de différentes fonctions qui permettent la communication entre le microcontrôleur et le système d'affichage LCD. Ce dernier se compose d'un contrôleur et d'un écran LCD. Deux fonctions permettent de communiquer avec l'afficheur LCD, SEND\_CMDE (Figure 5) et AFFI\_CAR (figure 6). La première réalise l'envoi de commandes à l'afficheur tandis que la seconde permet l'affichage d'un caractère ASCII.

La fonction WAY\_BUZY (Figure 7) permet d'attendre que l'afficheur soit prêt à recevoir une nouvelle transmission. Les fonctions CURS\_DEB (Figure 2) et CURS\_DEB\_L2 (figure 3) permettent de positionner le curseur en début de 1<sup>ère</sup> ou bien de 2<sup>ème</sup> ligne.

La fonction CURS\_OFF éteint le curseur (Figure 4 ). Enfin la fonction INIT\_LCD réalise l'initialisation de l'afficheur (Figure 1 ).

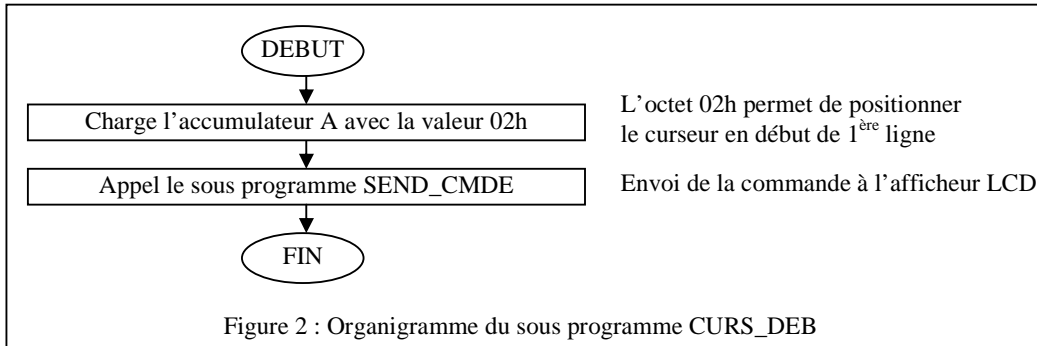
### 3.1 Sous programme INIT\_LCD (Figure 1 )

Ce sous programme réalise l'initialisation de l'afficheur LCD par l'envoi de commandes par l'intermédiaire de la fonction SEND\_CMDE.



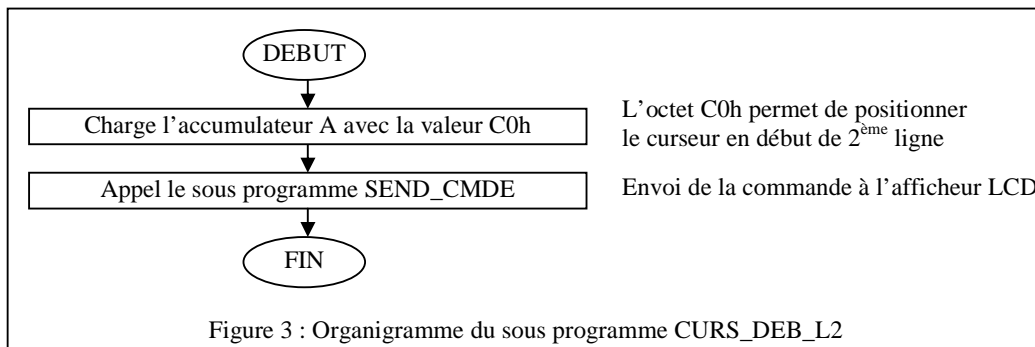
### 3.2 Sous programme CURS\_DEB (Figure 2 )

Le sous programme CURS\_DEB positionne le curseur à la position initiale, à savoir au début de la 1<sup>ère</sup> ligne. L'envoi de la commande se fait par la fonction SEND\_CMDE.



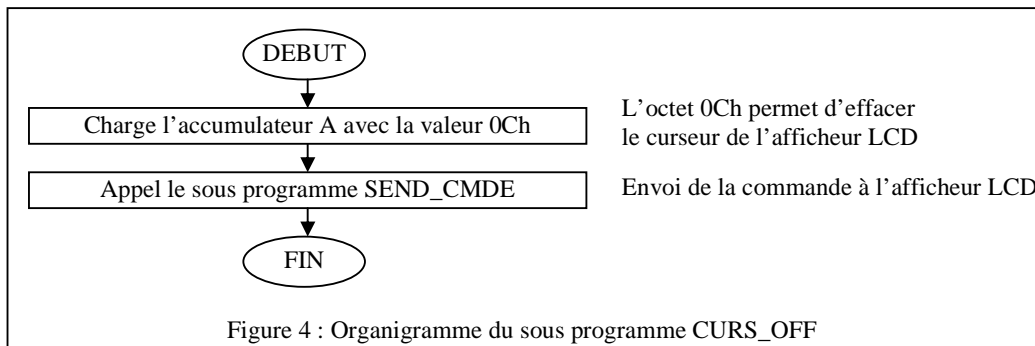
### 3.3 Sous programme CURS\_DEB\_L2 (Figure 3 )

Le sous programme CURS\_DEB\_L2 positionne le curseur à la position initiale, à savoir au début de la 2<sup>ème</sup> ligne. L'envoi de la commande se fait par la fonction SEND\_CMDE.



### 3.4 Sous programme CURS\_OFF (Figure 4 )

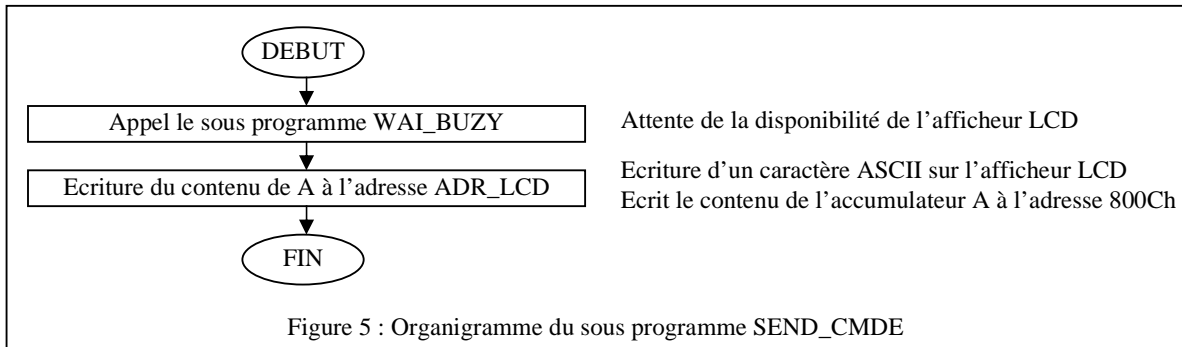
Le sous programme CURS\_OFF désactive le curseur. L'envoi de la commande se fait par la fonction SEND\_CMDE.



### 3.5 Sous programme SEND\_CMDE

(Figure 5 )

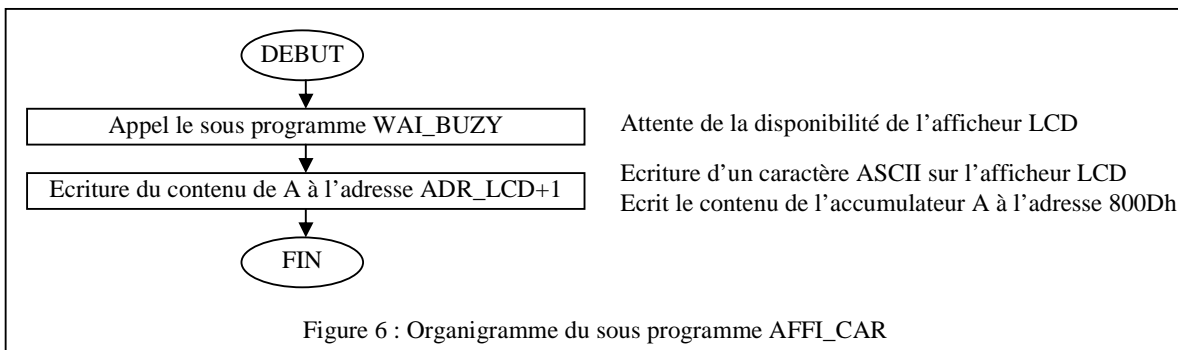
Le sous programme SEND\_CMDE envoie la commande contenue dans l'accumulateur A à l'afficheur LCD après vérification de la disponibilité de celui-ci par la fonction WAI\_BUZY.



### 3.6 Sous programme AFFI\_CAR

(Figure 6 )

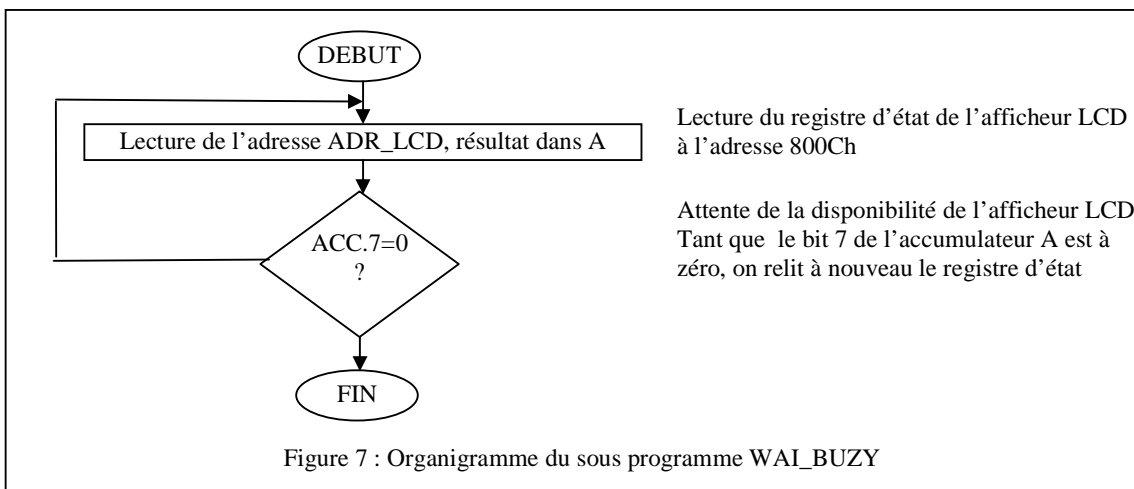
Le sous programme AFFI\_CAR envoie le caractère ASCII contenu dans l'accumulateur A à l'afficheur LCD après vérification de la disponibilité de celui-ci par la fonction WAI\_BUZY.



### 3.7 Sous programme WAI\_BUZY

(Figure 7 )

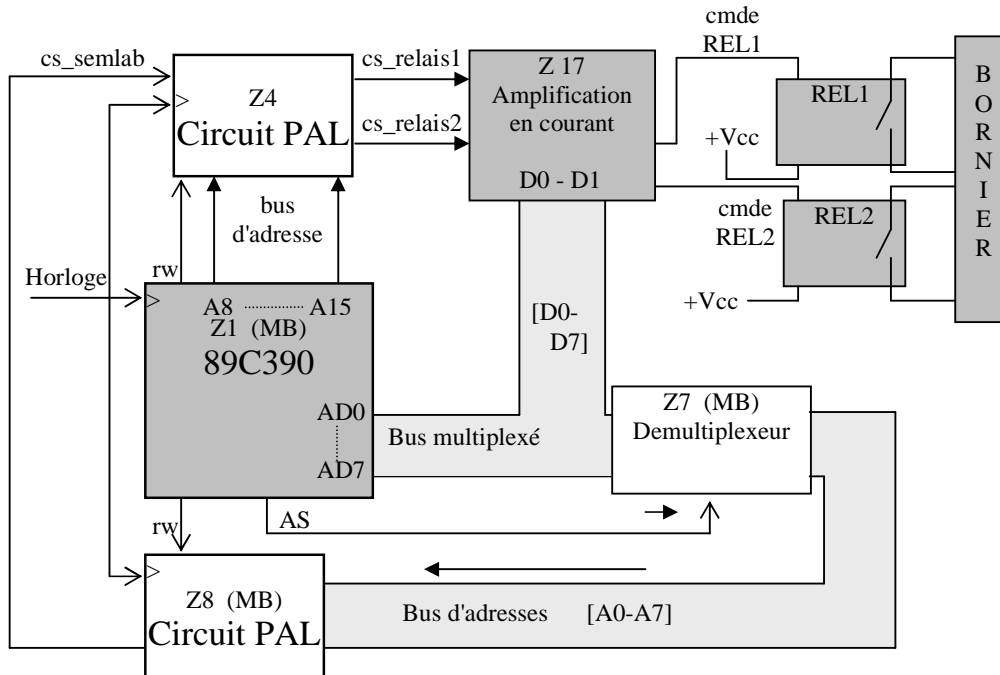
Le sous programme WAI\_BUZY attend que l'afficheur soit prêt à recevoir une information.



### III - Sorties relais (REL1 et REL2)

ATTENTION : l'interrupteur SW1 doit être en position 2 en cas d'utilisation du relais 1. (La commande du relais 1 est également utilisée par la sortie thermostat du DS1621, SW1 en position 1).

#### 1. Liaison microcontrôleur vers les sorties relais



(MB) : Composants se trouvant sur la carte mère MC51TT  
Les autres composants se trouvent sur la carte SemLab1-B

#### 2. Descriptif du fonctionnement

Le système de commande de relais est composé de deux relais dont les adresses accessibles en écriture sont les suivantes : 8018h pour REL1 et 8019h pour REL2. Il dispose également de deux sorties "tout ou rien" accessibles sur le bornier, commandées par les relais.

Il est possible de visualiser l'état des deux relais grâce aux deux leds, LD1 et LD2, qui ont été prévues à cet effet. La commande des relais s'effectue par le positionnement du bit D0 du bus de données lors d'un accès aux adresses 8018h ou 8019h (Tableau 1).

Tableau 1 : Adressage des relais

Bus de données :	D7	D6	D5	D4	D3	D2	D1	D0
8018h :	X	X	X	X	X	X	X	rel1
8019h :	X	X	X	X	X	X	X	rel2

Le tableau 2 montre la correspondance entre le niveau logique du bit de commande des relais et l'état des leds.

Tableau 2 : visualisation de l'état des relais au moyen des leds LD1 et LD2

Etat de la commande	Etat des leds	Circuit électrique équivalent
rel1/2 = '1'	LD1/2 : éteinte	REL1/2 = circuit ouvert
rel1/2 = '0'	LD1/2 : allumée	REL1/2 = circuit fermé

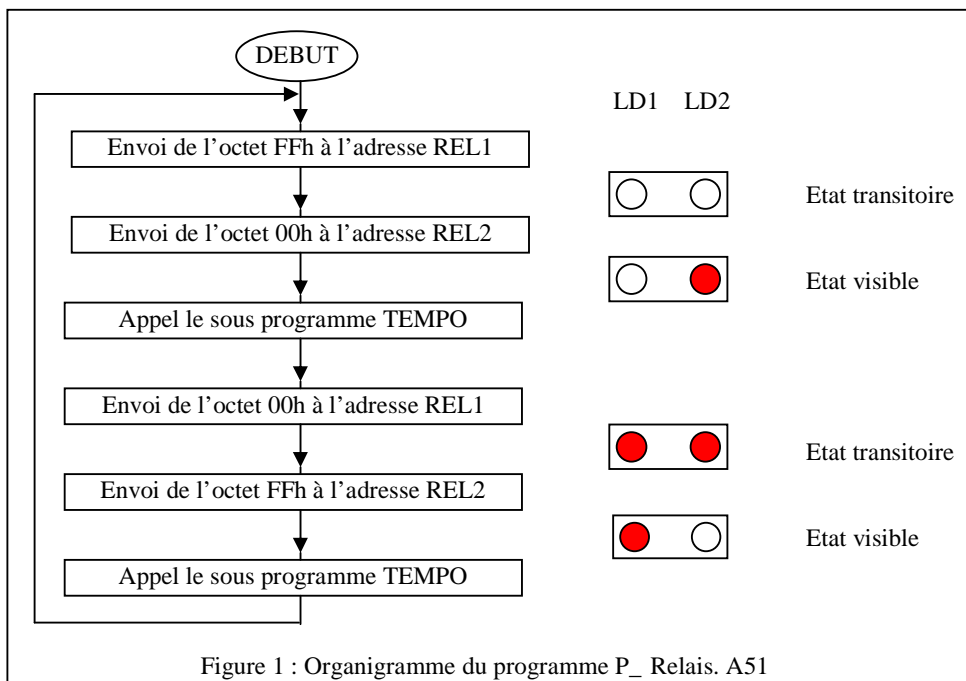
Rappel : le montage électrique équivalent entre les bornes des sorties  
REL1 ou REL2 est un interrupteur. Pour les utiliser, il faut les relier à une  
alimentation à travers une charge.

→ Attention à ne pas court-circuiter les sorties.

### 3. Description du programme P\_Relais.A51

Dans une boucle infinie, on fait des écritures successives aux adresses REL1 et REL2 de manière à allumer et à éteindre alternativement les leds LD1 et LD2.

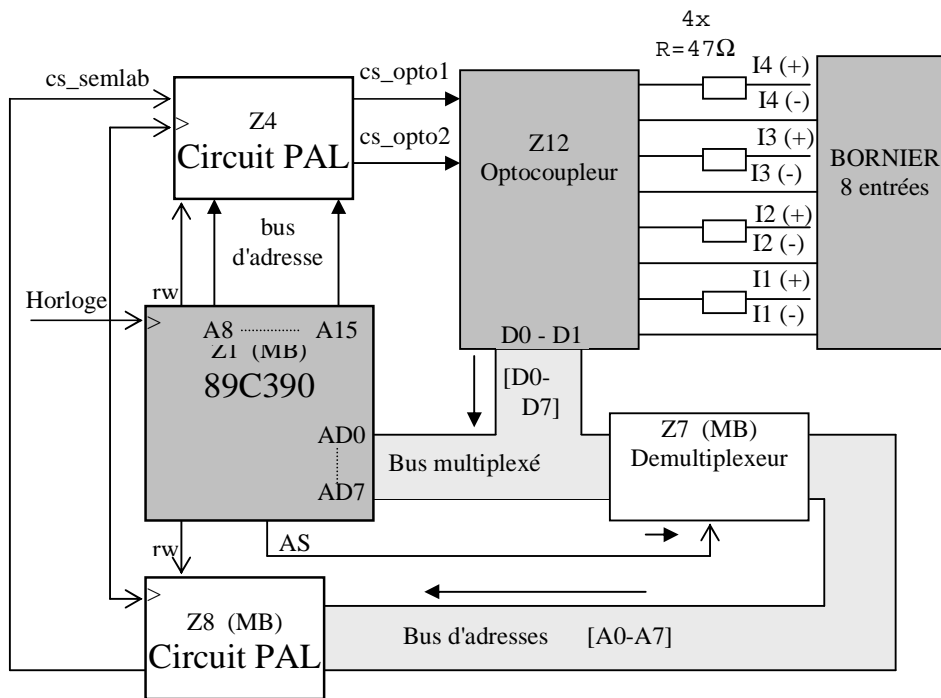
#### 3.1 Organigramme du programme principal (figure 1)



## IV - Entrées optocouplées (TIL199)

ATTENTION : l'interrupteur SW1 doit être en position 2 en cas d'utilisation du relais 1. (La commande du relais 1 est également utilisée par la sortie thermostat du DS1621, SW1 en position 1).

### 1. Liaison microcontrôleur vers les entrées optocouplées



(MB) : Composants se trouvant sur la carte mère MC51TT  
Les autres composants se trouvent sur la carte SemLab1-B

### 2. Descriptif du fonctionnement

Le circuit TIL199 dispose de quatre entrées optocouplées nommées I1, I2, I3 et I4 et qui sont accessibles sur le bornier.

La lecture des entrées optocouplées s'effectue à partir des bits D1 et D0 du bus de données lors d'un accès aux adresses 8018h ou 8019h ( Tableau 1 ).

Tableau 1 : Adressage de l'optocoupleur

Bus de donn�e :	D7	D6	D5	D4	D3	D2	D1	D0
8018h :	X	X	X	X	X	X	I2	I1
8019h :	X	X	X	X	X	X	I4	I3

Si l'on entre une différence de potentiel non nulle en respectant la polarité sur l'une des entrées optocouplées, on récupère un niveau '1' logique en sortie de l'optocoupleur, et un niveau '0' logique dans le cas contraire.

### 3. Description du programme P\_RelaisOpto.A51

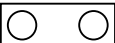



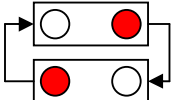
Ce programme permet de lire l'état des quatre entrées analogiques et de le visualiser sur les leds LD1 et LD2 qui indiquent respectivement l'état des relais REL1 et REL2 . L'utilisation des relais est présentée dans le chapitre précédent.

La variable *clign* a été déclarée en mémoire externe mais il est toutefois possible de la déclarer en mémoire interne sauf qu'il est préférable de ne pas utiliser la mémoire interne car elle est beaucoup occupée par des fonctions du moniteur

Dans une boucle infinie, on fait une lecture des entrées optocouplées puis on allume les leds comme indiqué dans le tableau 2. Dans le cas où l'entrée I4 serait active, les leds clignoteraient grâce à la variable *Clign* qui permet de mémoriser l'état précédent des leds de façon à le compléter lors de chaque passage dans la boucle.

Le programme P\_RelaisOpto.A51 représente un cycle de lecture des entrées optocouplées, de traitement des résultats puis d'écriture sur les sorties relais.

Tableau 2 : Etat des leds en fonction de l'entrée optocouplée active

Etat des entrées optocouplées	Etat des leds LD1 et LD2
Aucune entrée active	
Entrée I1 active	
Entrée I2 active	
Entrée I3 active	
Entrée I4 active	

### 3.1 Organigramme du programme ( Figure 1 )

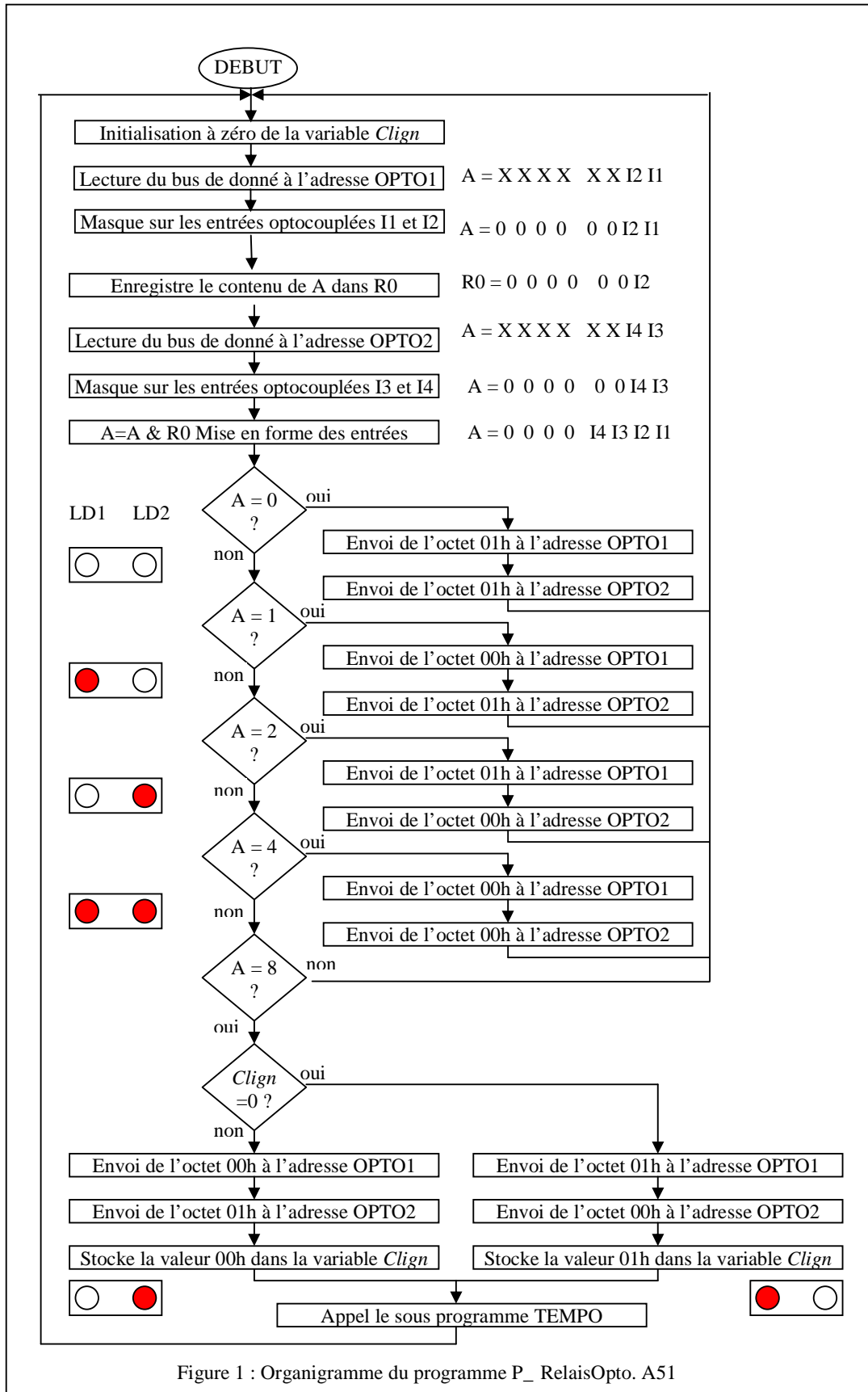
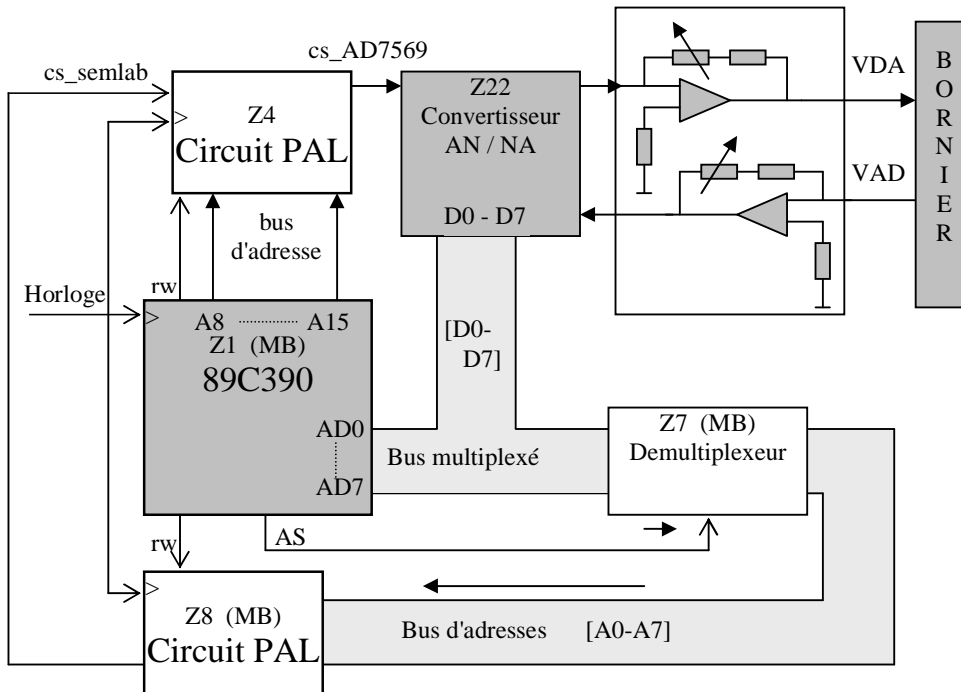


Figure 1 : Organigramme du programme P\_ RelaisOpto. A51

## V - Convertisseurs AD et DA rapides AD7569 (Z22)

### 1. Liaison microcontrôleur vers le convertisseur AD/DA



(MB) : Composants se trouvant sur la carte mère MC51TT  
Les autres composants se trouvent sur la carte SemLab1-B

### 2. Descriptif du fonctionnement

Le circuit AD7569 contient un convertisseur analogique/digital et un convertisseur digital/analogique (notés VAD et VDA sur le connecteur). Les tensions sont limitées entre +2,5 volts et -2,5 volts. L'entrée est protégée par des diodes de limitation.

Des amplificateurs opérationnels montés en inverseur permettent d'amplifier ou d'atténuer les signaux. Ce qui implique donc que les valeurs lues seront opposées aux valeurs réelles

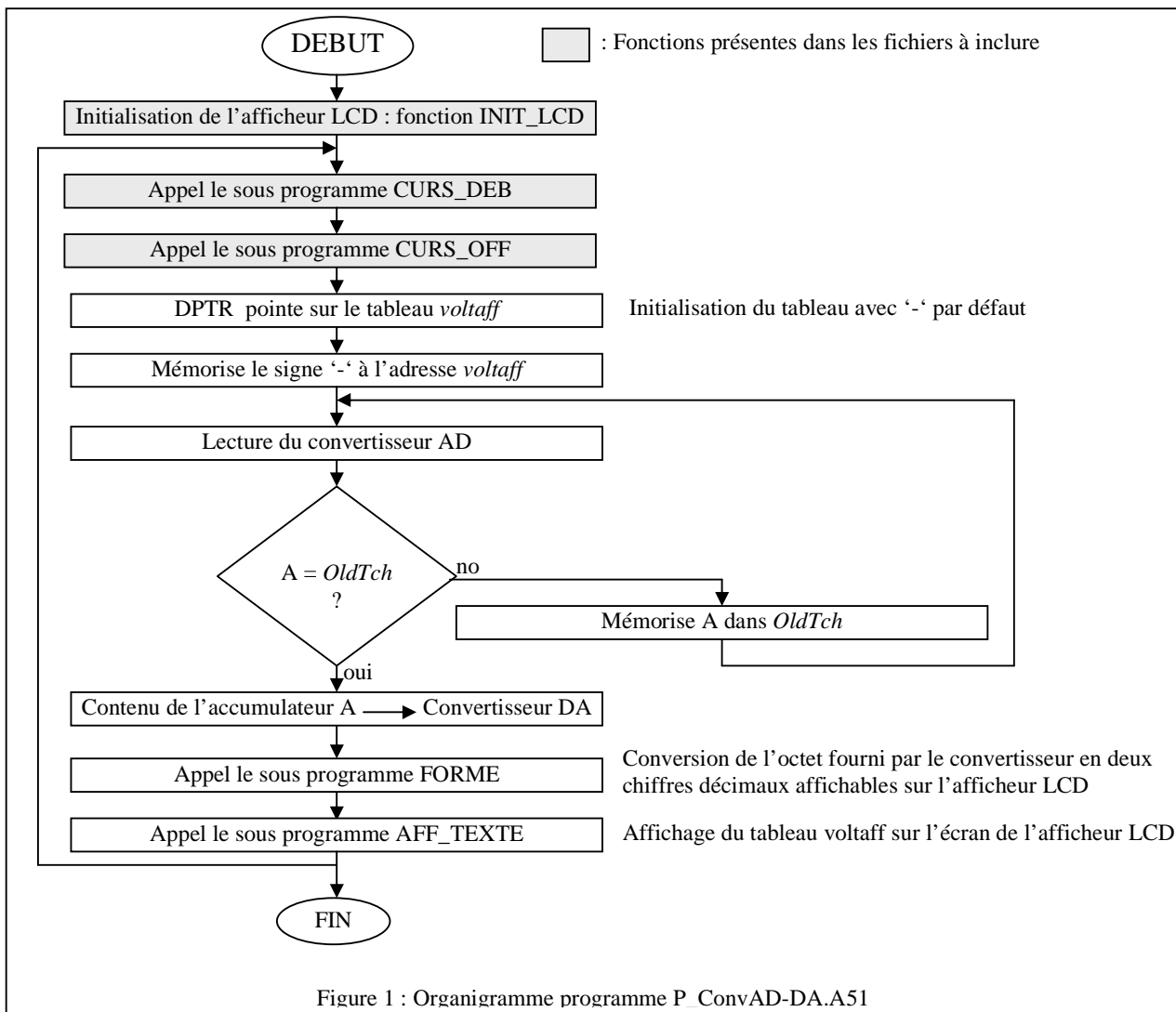
Les temps de conversion sont de 1  $\mu$ s en DA et 2  $\mu$ s en AD. Il faudra donc prendre en compte la durée écoulée entre deux conversions. Le convertisseur est adressé en 8000h. Une écriture à cette adresse initialise le convertisseur DA avec la valeur écrite. Une lecture initialise le convertisseur AD et la valeur lue est le résultat de la précédente conversion.

Les valeurs peuvent être inexactes lors des premiers tests car il faut régler les deux potentiomètres situés à côté des ports VDA et VAD.

### 3. Description du programme P\_ConvAD-DA.A51

Dans une boucle infinie, on fait une lecture de la tension sur VAD pour l'afficher sur l'écran de l'afficheur LCD

#### 3.1 Organigramme du programme principal ( Figure 1 )



Ce programme, dans une boucle infinie, réalise la conversion du signal analogique présent sur son entrée, met en forme cette valeur ainsi obtenue qui est stockée dans le tableau *voltaff* ( Tableau 1 ) puis l'affiche sur l'écran de l'afficheur LCD.

Tableau 1 : Contenu du tableau *voltaff*

Tableau <i>voltaff</i> pointé par DPTR	DPTR	DPTR + 1	DPTR + 2	DPTR + 3	DPTR + 4
Contenu du tableau <i>voltaff</i> à l'initialisation	-	0	.	0	V

### 3.2 Sous programmes INIT\_LCD, CURS\_DEB, CURS\_OFF et AFFI\_CAR

Ces quatre sous programmes se trouvent dans le fichier à inclure S\_AFILCD.A51 et sont présentés dans le chapitre traitant de l'afficheur LCD.

### 3.3 Sous programme FORME (Figure 2)

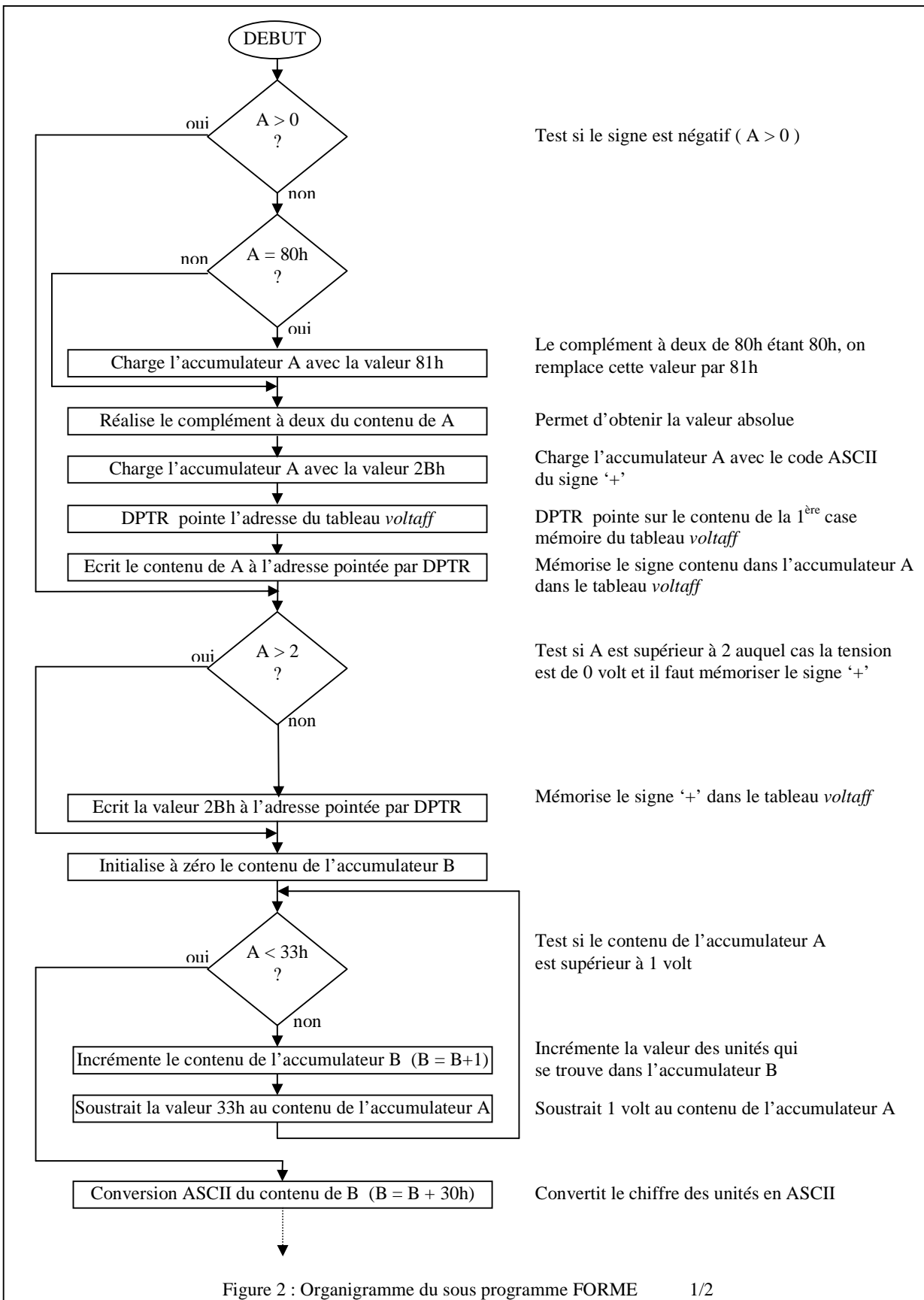


Figure 2 : Organigramme du sous programme FORME

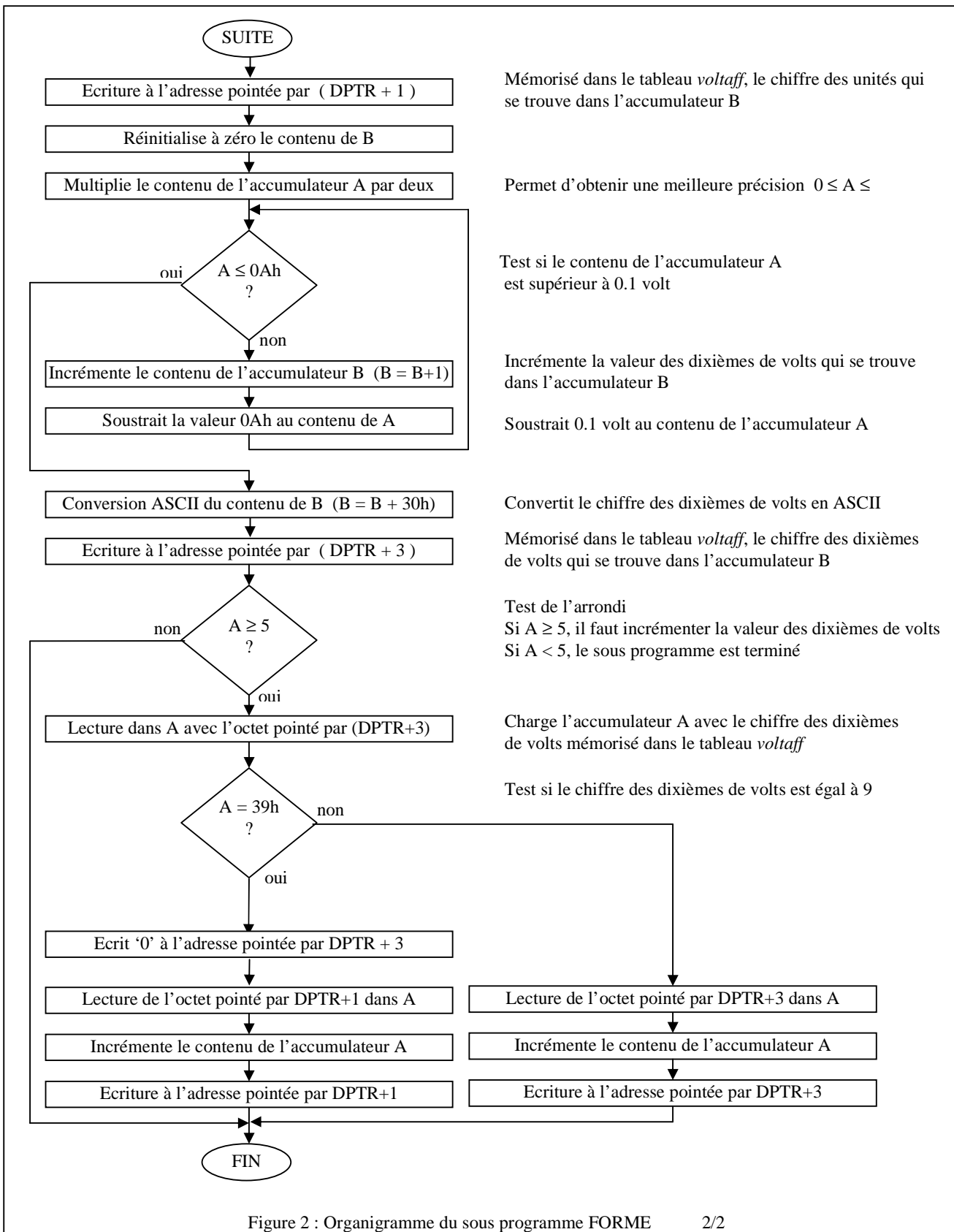
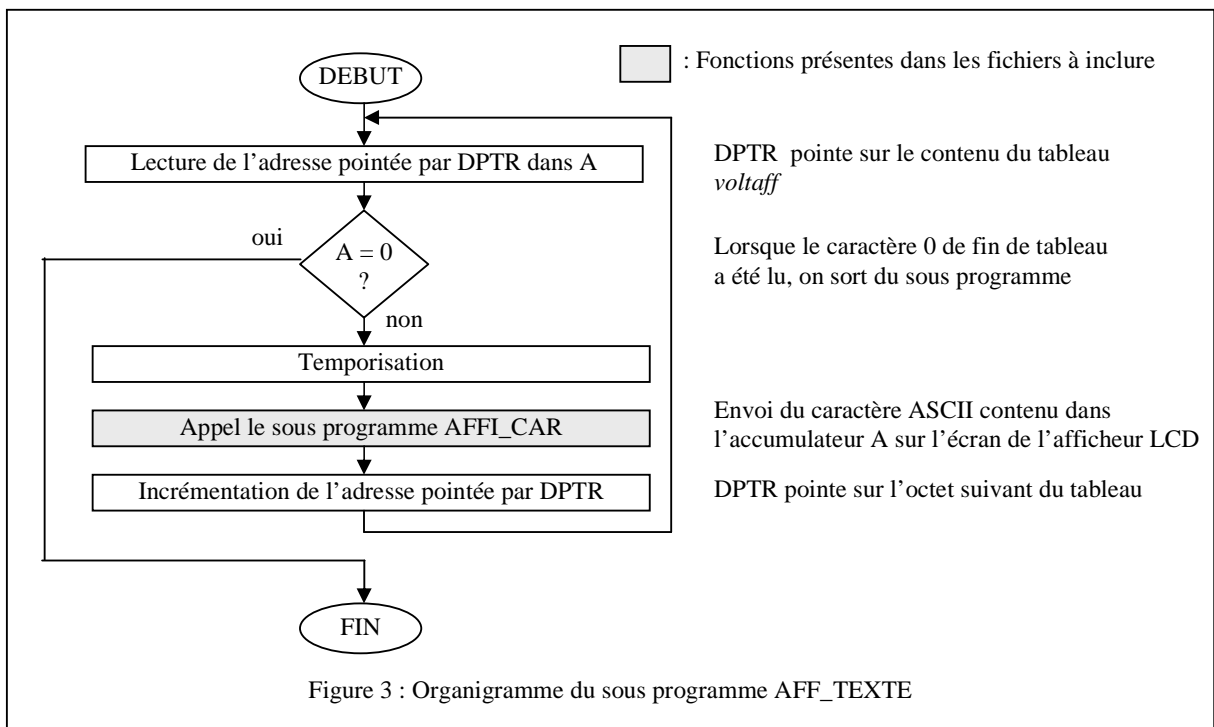


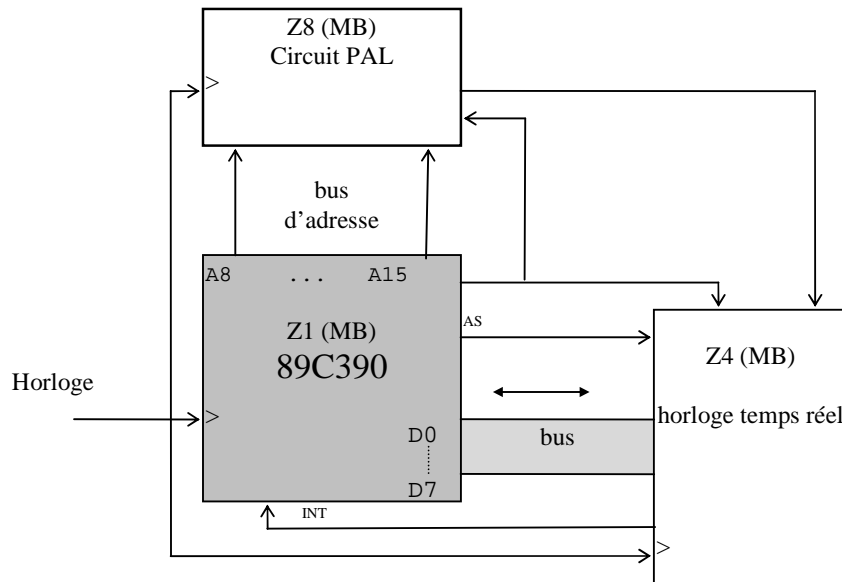
Figure 2 : Organigramme du sous programme FORME

3.4 Sous programme AFF\_TEXTE (Figure 3 )



# VI - HORLOGE TEMPS REEL (DS-12887)

## 1. Liaison microcontrôleur vers le circuit RTC - horloge temps réel



## 2. Descriptif du fonctionnement

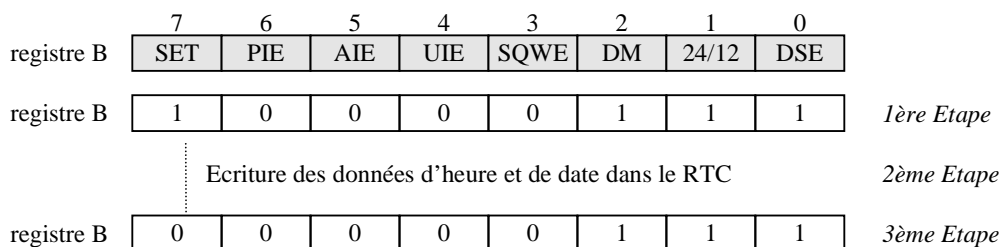
### Plan de l'espace mémoire

ADR_RTC	Adresse	Contenu
8200h	00	secondes
	01	alarme secondes
	02	minutes
	03	alarme minutes
	04	heures
	05	alarme heures
	06	jour de la semaine
	07	jour du mois
	08	mois
	09	année
820Ah	0A	REGISTRE A
	0B	REGISTRE B
	0C	REGISTRE C
	0D	REGISTRE D
	0E-3F	50 octets libres utilisateur

### 3. Descriptif du programme P\_INIT\_HTR.A51

Rôle: Fichier d'initialisation du composant R.T.C. (Real Time Clock). Les paramètres concernant l'heure, la date et l'alarme du circuit RTC peuvent être modifiés.

Pour initialiser l'heure et la date du composant DS12887, il faut positionner des bits dans le registre B.



⇒ *Première Etape*

- bit *SET* = '1'            Le composant n'effectue plus de mise à jour des données de date et d'heure.
- bits *24/12 / DSE* = '1'        format horaire 24 heures / mise à jour de l'heure pour le passage heure d'été/hiver
- bit *DM* = '0'            les données sont de type BCD

⇒ *Deuxième Etape*

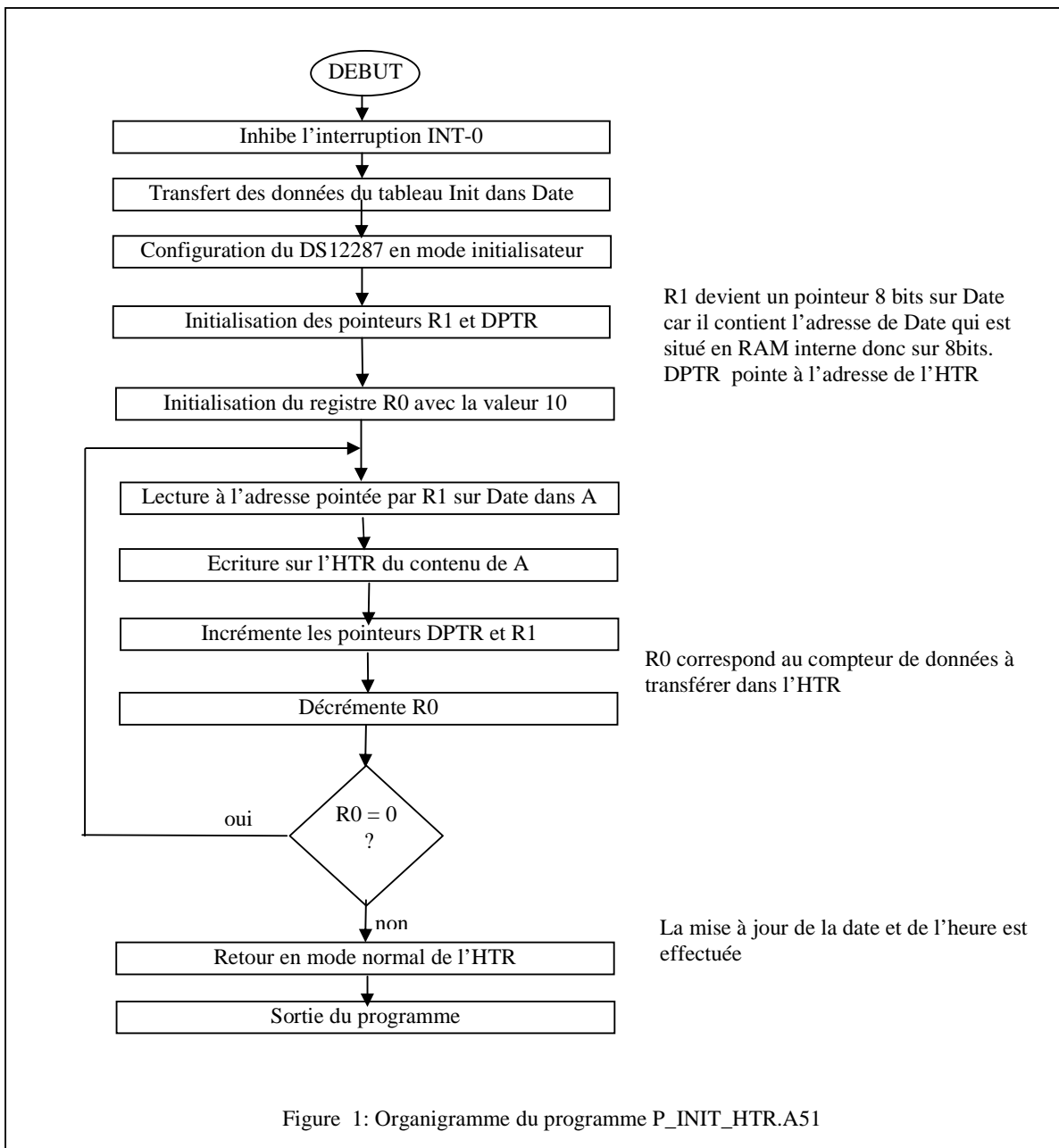
Ecrire les données (heure et date) en mémoire du circuit RTC  
(adresse 8200h à 8209h)

⇒ *Troisième Etape*

Replacer le bit *SET* à '0' Fonctionnement normal du composant  
mise à jour des données de date et d'heure chaque seconde

avec une

### 3.1 Organigramme du programme P\_INIT\_HTR.A51 (figure 1)



### 4. Descriptif du programme P\_Htr\_IT.a51

Rôle: Afficher sur l'afficheur LCD les données issues du composant R.T.C. (Heure, Minute, Seconde, date).

#### 4.1 Configuration du circuit RTC

Avant d'utiliser le composant RTC, il faut s'assurer de son bon fonctionnement et de le configurer.

⇒ Lecture du registre D pour vérifier si le circuit RTC est fonctionnel

	7	6	5	4	3	2	1	0
registre D	VRT	0	0	0	0	0	0	0

Si le bit 7 de ce registre est à '0', un problème matériel du circuit RTC est à envisager

⇒ Ecriture dans le registre B pour configurer le circuit RTC

	7	6	5	4	3	2	1	0
registre B	SET	PIE	AIE	UIE	SQWE	DM	24/12	DSE
registre B	0	0	0	1	0	1	1	1

L'écriture de cet octet permet de placer le composant en mode normal (bit SET='0'), autorise les interruptions de mise à jour chaque seconde (bit UIE='1').

Les bits 0, 1 et 2 permettent de sélectionner un format de donnée de type BCD, un format horaire de 24 heures et une mise à jour des données du composant lors du passage heure d'été/ heure d'hiver.

⇒ Ecriture dans le registre A pour déclencher l'oscillateur interne du RTC

	7	6	5	4	3	2	1	0
registre A	UIP	DV2	DV1	DV0	RS3	RS2	RS1	RS0
registre A	1	0	1	0	0	0	0	0

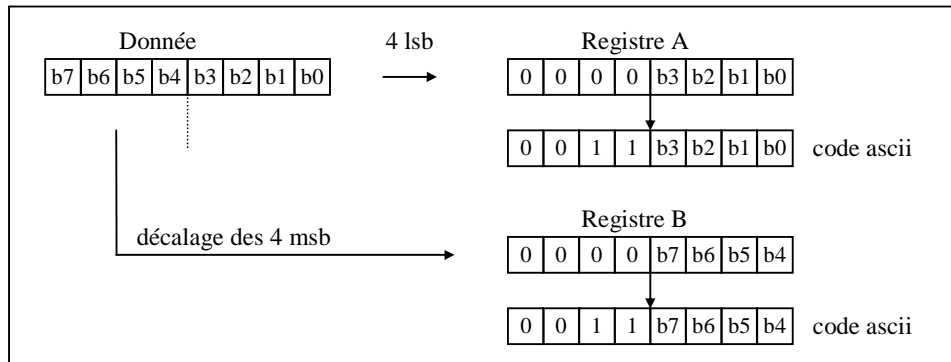
Le bit UIP='1' permet d'obtenir une mise à jour des données (heure et date).

Les bits 6, 5 et 4 = '010', ce code permet de déclencher l'oscillateur interne du RTC.

## 4.2 Déroulement du programme

A chaque interruption de mise à jour des données (chaque seconde), une interruption INT-0 est générée par le composant. Le traitement de cette interruption consiste en l'affichage des données du composant :

⇒ Affichage des heures, minutes et secondes .Ces 3 données de type BCD sont respectivement situées aux adresses 8204h, 8202h et 8200h. Avant de pouvoir les afficher sur l'écran LCD, il faut effectuer une conversion BCD -> ASCII



La donnée (code BCD) est divisée entre les registres A et B du microcontrôleur.

Le registre A contient les unités de la donnée à afficher. Il faut ajouter 30h à ce chiffre pour obtenir le code ASCII correspondant.

Le registre B contient les dizaines de la donnée à afficher. Il faut ajouter 30h à ce chiffre pour obtenir le code ASCII correspondant.

⇒ Affichage de la date

Les données concernant la date sont situées aux adresses

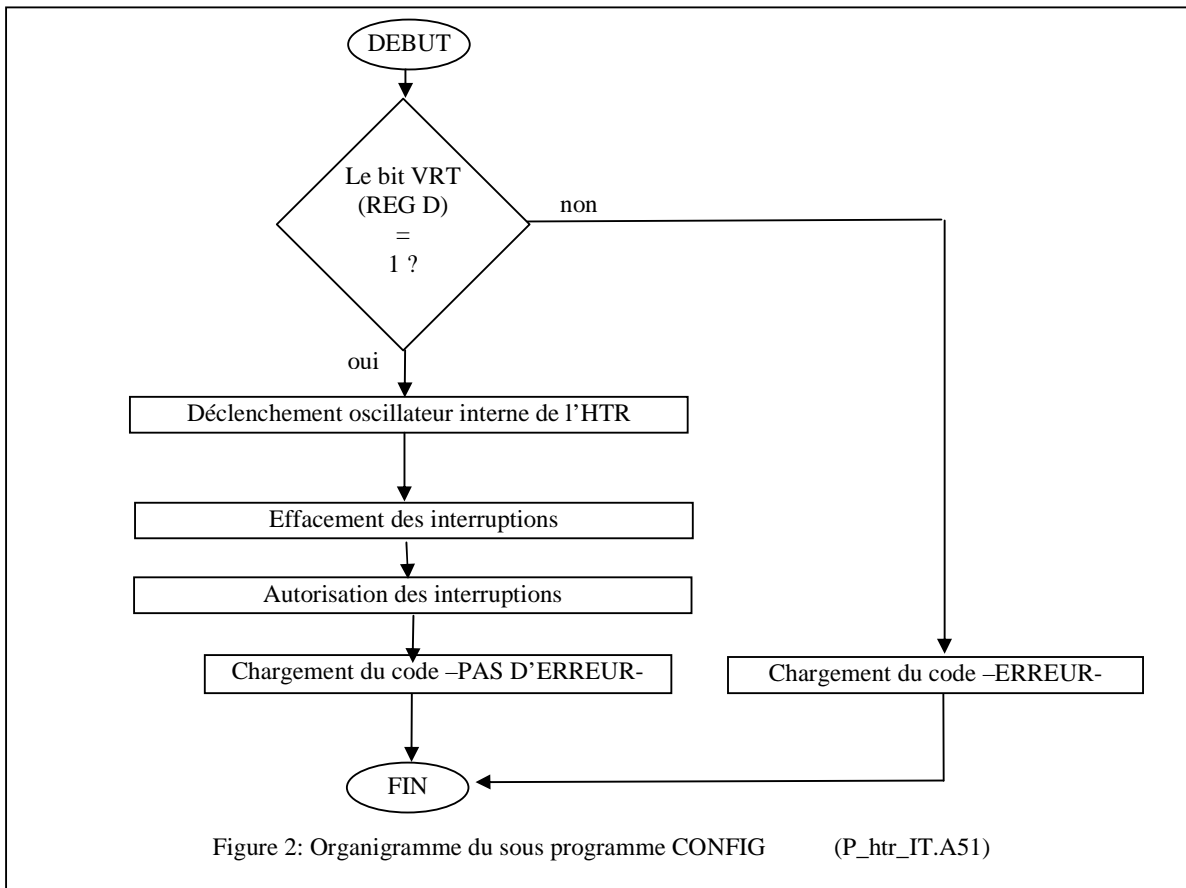
8206h (jour de la semaine), 8207h (jour du mois), 8208h (mois) et 8209h (année).

En fin de routine d'interruption, il faut replacer le bit UF du registre C à '0' pour pouvoir redétecter une interruption de mise à jour des données.

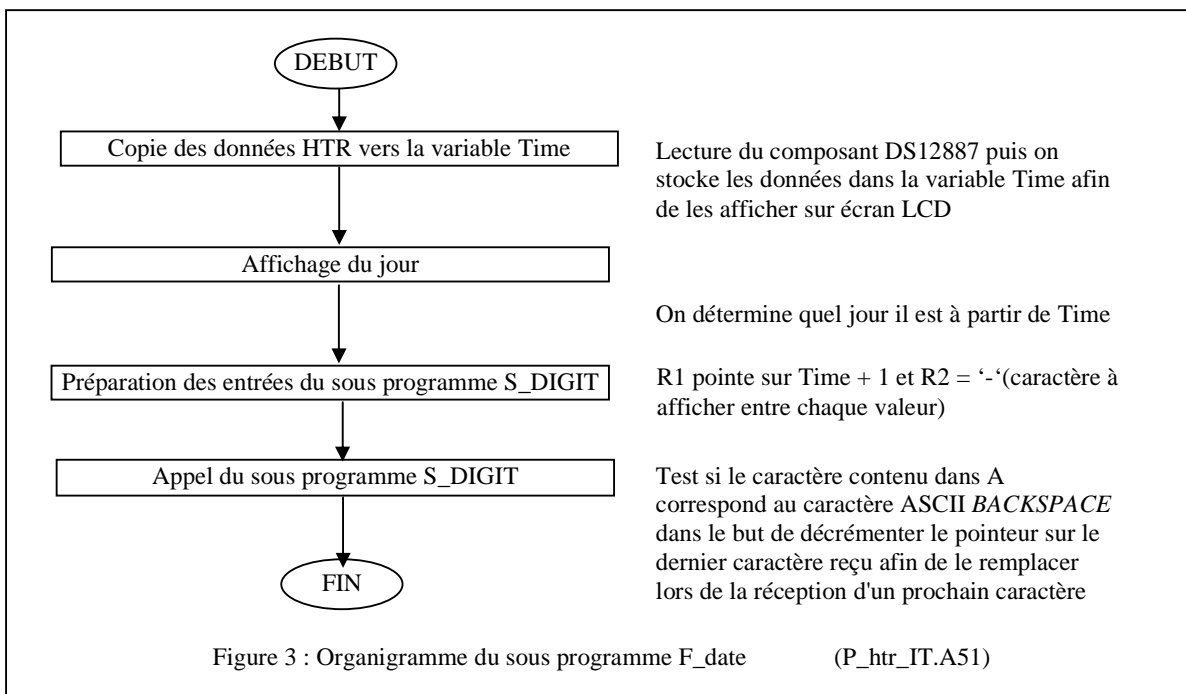
Pour remettre à '0' ce bit, il faut effectuer une lecture du registre C.

Le programme S\_AFILCD.a51 contient les initialisations de l'écran LCD (voir partie écran LCD).

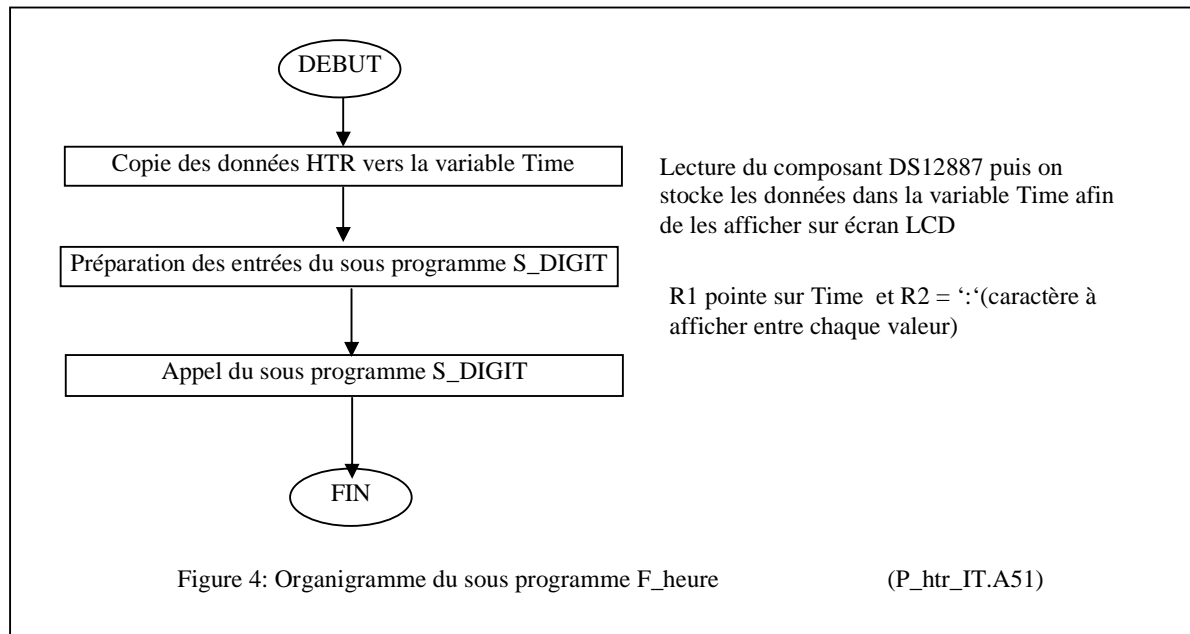
### 4.3 Organigramme du sous programme CONFIG (figure 2)



### 4.4 Organigramme du sous programme F\_date (figure 3)



#### 4.5 Organigramme du sous programme F\_heure (figure 4)



#### 4.6 Organigramme du sous programme S\_digit (figure 5)

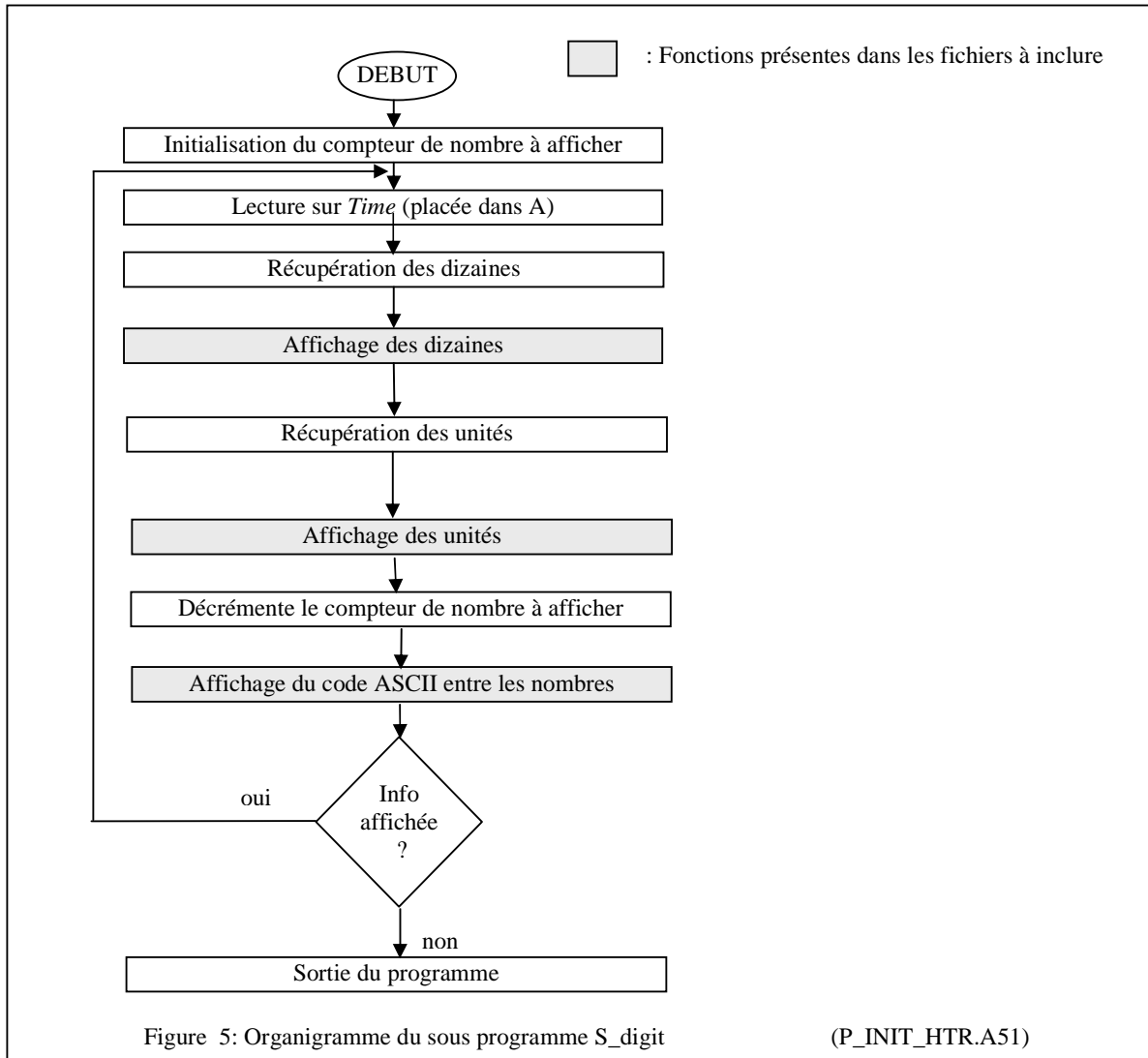
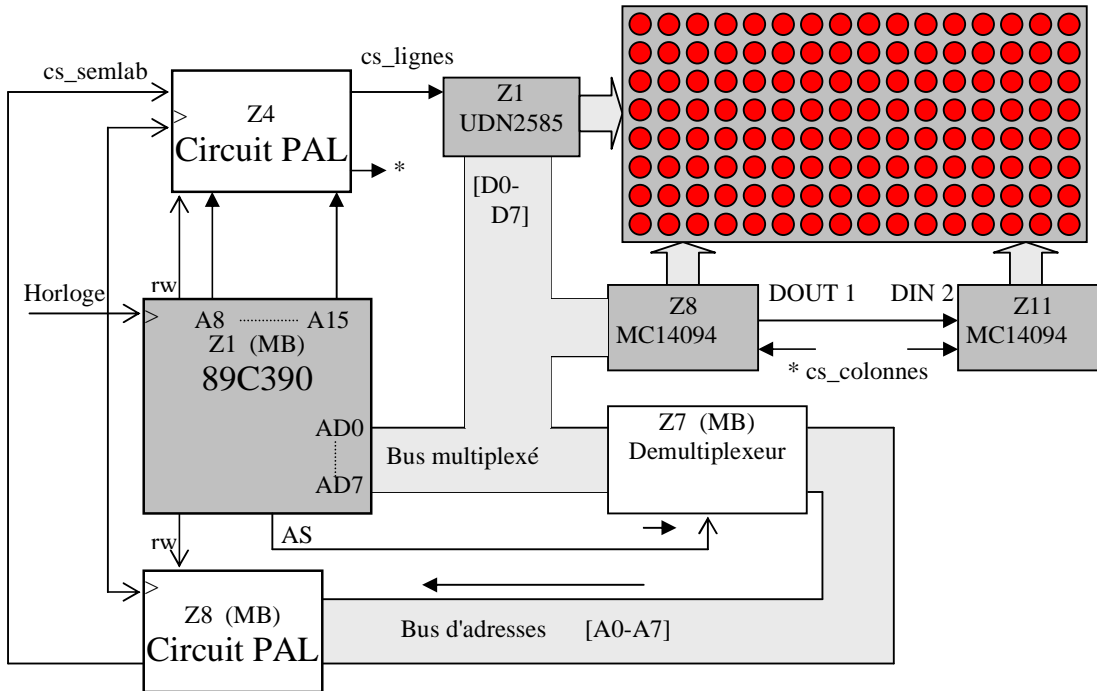


Figure 5: Organigramme du sous programme S\_digit

(P\_INIT\_HTR.A51)

## VII - PANNEAU DE LED

### 1. Liaison microcontrôleur vers le panneau de leds



(MB) : Composants se trouvant sur la carte mère MC51TT  
Les autres composants se trouvent sur la carte SemLab1-B

### 2. Descriptif du fonctionnement

Le panneau d'affichage est organisé en 8 lignes de 16 leds. Chaque sortie du circuit UDN-2585 (Z1) commande les 8 lignes de leds. Les entrées de ce circuit se situent à l'adresse 8008h.

Les 16 colonnes sont commandées, à travers des résistances en série, par les sorties des deux circuits ULN-2803 (Z6 et Z10).

Les entrées de ces circuits sont commandées par deux registres à décalage 4094 (Z8 et Z11).

Avec le chaînage de la sortie série d'un registre à décalage sur l'entrée du registre suivant, les deux registres sont vus comme un seul registre de 16 bits.

Lorsqu'on écrit à l'adresse 8004h, les bits de poids fort du bus de données sont utilisés pour piloter les registres à décalage (Tableau 1).

Le signal "Chip Select" à cette adresse sert d'horloge de décalage.

Tableau 1 : Configuration du registre à décalage

Bus de données	B7	B6	B5	B4	B3	B2	B1	B0
Octet à écrire dans le registre à décalage	OENB	STRB	LOAD	DATA	X	X	X	X

Bit OENB : Active les sorties du registre à décalage.

Bit STRB : Transfère le contenu des latches du registre vers les sorties.

Bit LOAD : Commande de latcher le bit DATA.

Bit DATA : Donnée sur 1 bit à entrer dans le premier latch du registre à décalage.

Pour allumer une led lxn (led de la colonne x, ligne n), il faut :

- mettre le bit n du circuit UDN-2585 à 0
- mettre le bit x (de 0 à 15) du registre à décalage à 1
- mettre le signal OENB à 1

## PRINCIPE DE L'AFFICHAGE

L'affichage se fait par multiplexage temporel des 8 lignes du panneau.

A un instant t, on allume une ou plusieurs leds d'une seule ligne.

Après une certaine temporisation on passe à la ligne suivante.

Pendant la temporisation d'une ligne, on peut modifier le contenu du registre à décalage avec la configuration de la ligne suivante. Durant cette modification, le signal STRB doit être à 0 et OENB à 1.

### 1 - Configurer les 16 bits du registre des colonnes.

a - mettre DATA à 0 ou 1 suivant la valeur de la colonne, LOAD à 1 et STRB à 0 et OENB inchangé par rapport à l'état précédent.

b - écrire cette valeur à l'adresse 8004h.

Répéter a) et b) autant de fois que de colonnes à configurer.

La 1ère donnée entrée dans le registre correspond à la dernière colonne à configurer.

### 2 - Sélectionner la ligne à afficher.

Ecrire à l'adresse 8008h un octet avec le bit de la ligne à allumer à 0 et tous les autres bits à 1. Par exemple la valeur 0FEh allume la 1ère ligne (en haut du panneau) ; la valeur 0BFh allume la 7ème et avant dernière ligne.

### 3 - Activer les sorties du registre à décalage.

Mettre les deux signaux STRB et OENB à 1. Ceci active le transfert de la nouvelle configuration des latches du registre vers les sorties.

### 4 - Marquer une temporisation pour assurer la rémanence.

### 5 - Répéter 1, 2, 3 et 4 pour la ligne suivante.

Il y a deux types de message à afficher :

1 – Les caractères que l'on souhaite faire défiler seul sur le pavé de LED (Appel direct du sous programme DEFI\_LED avec l'adresse pointée sur le tableau contenant l'état des colonnes correspondant au caractère que l'on souhaite faire défiler). Les noms des variables déclarées dans le fichier «s\_sprled.a51» se terminent par 'sl' comme novalsl...

2 – Les chaînes de caractères (Appel tout d'abord les sous programmes CONV\_LED avant d'appeler le sous programme DEFI\_LED).

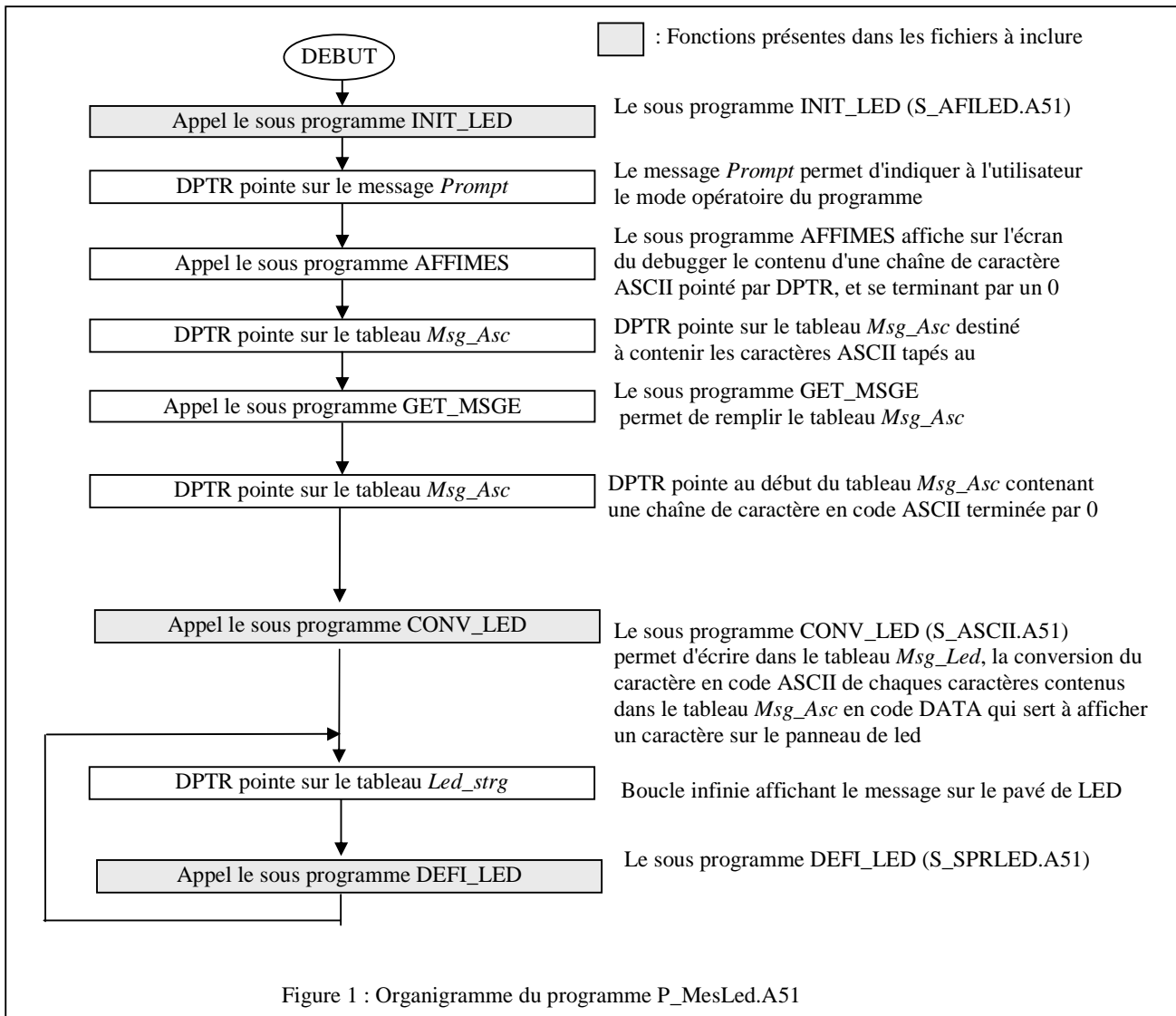
Voir les fichiers P\_MesLed.A51, S\_ASCII.A51, S\_Sprled.a51 et S\_AFILED.A51.

### 3. Descriptif du programme P\_MesLed.A51

#### 3.1 Organigramme du programme principal (Figure 1)

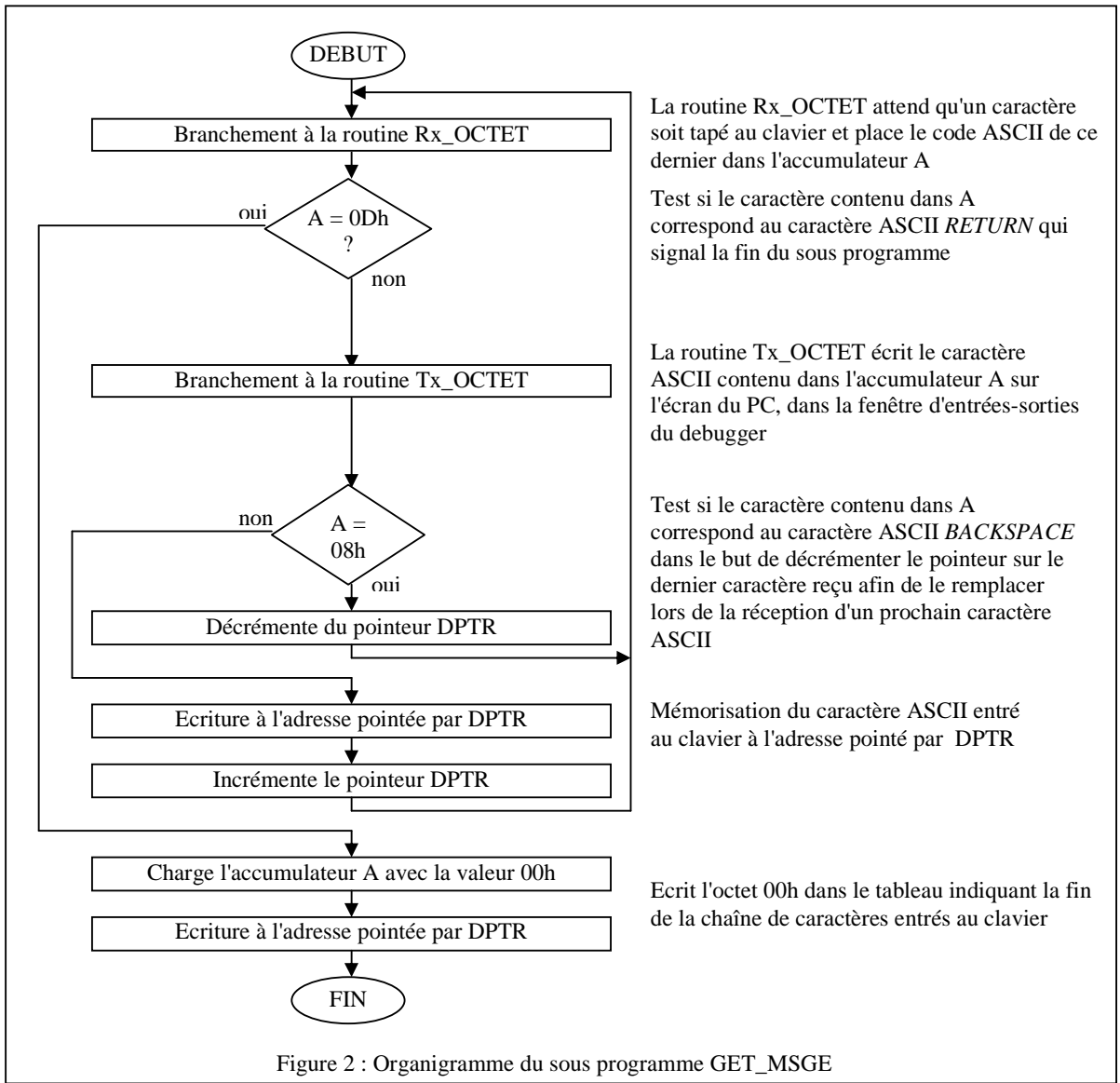
Le programme P\_MesLed.A51 commence par afficher un message Prompt dans la fenêtre d'entrées-sorties du débogueur indiquant à l'utilisateur la manière de l'exécuter. Il permet ensuite de lire un message tapé au clavier dont l'écho est transmis simultanément à la fenêtre d'entrées-sorties du débogueur puis de faire défiler le message complet sur le panneau de led.

Le programme utilise pour cela deux routines qui sont intégrées au moniteur. La routine Rx\_OCTET permet de placer dans l'accumulateur A tout caractère tapé au clavier. La routine Tx\_OCTET affiche dans la fenêtre d'entrées-sorties du débogueur le caractère ASCII contenu dans l'accumulateur A.



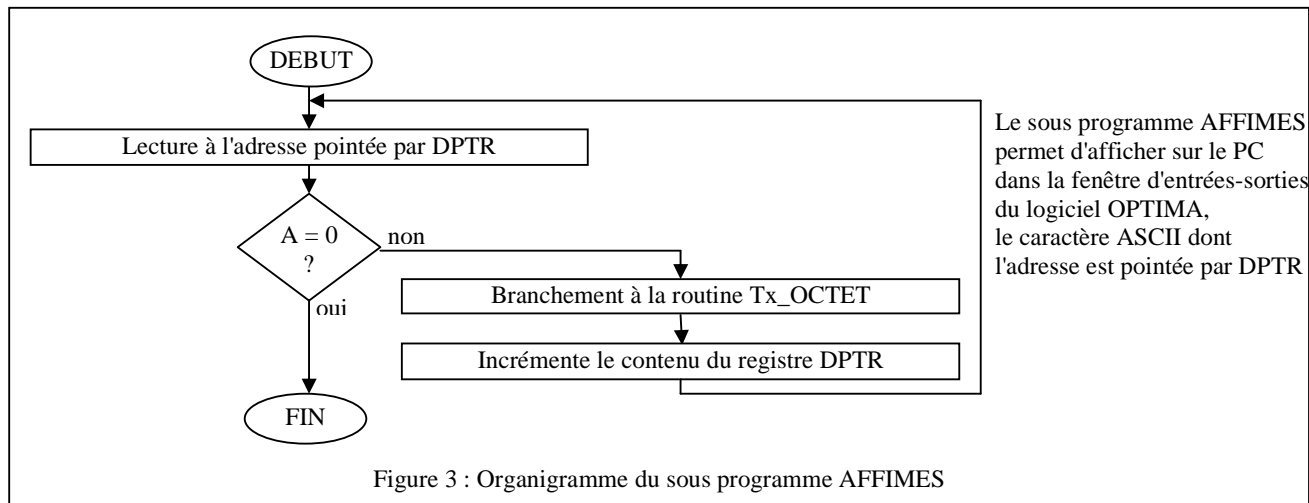
#### 3.2 Sous programme GET\_MSGE (Figure 2)

Le sous programme GET\_MSGE réalise la lecture des touches appuyées du clavier, leur mémorisation dans le tableau *Msg\_Led* pointé par DPTR, ainsi que de l'affichage de celles-ci sur la fenêtre d'entrées-sorties du débogueur.



### 3.3 Sous programme AFFIMES (Figure 3 )

Le sous programme AFFIMES affiche dans la fenêtre d'entrées-sorties du debugger la chaîne de caractères *Prompt* pointée par DPTR et se terminant par le chiffre 0. Le prompt indique la manière d'utiliser le programme.

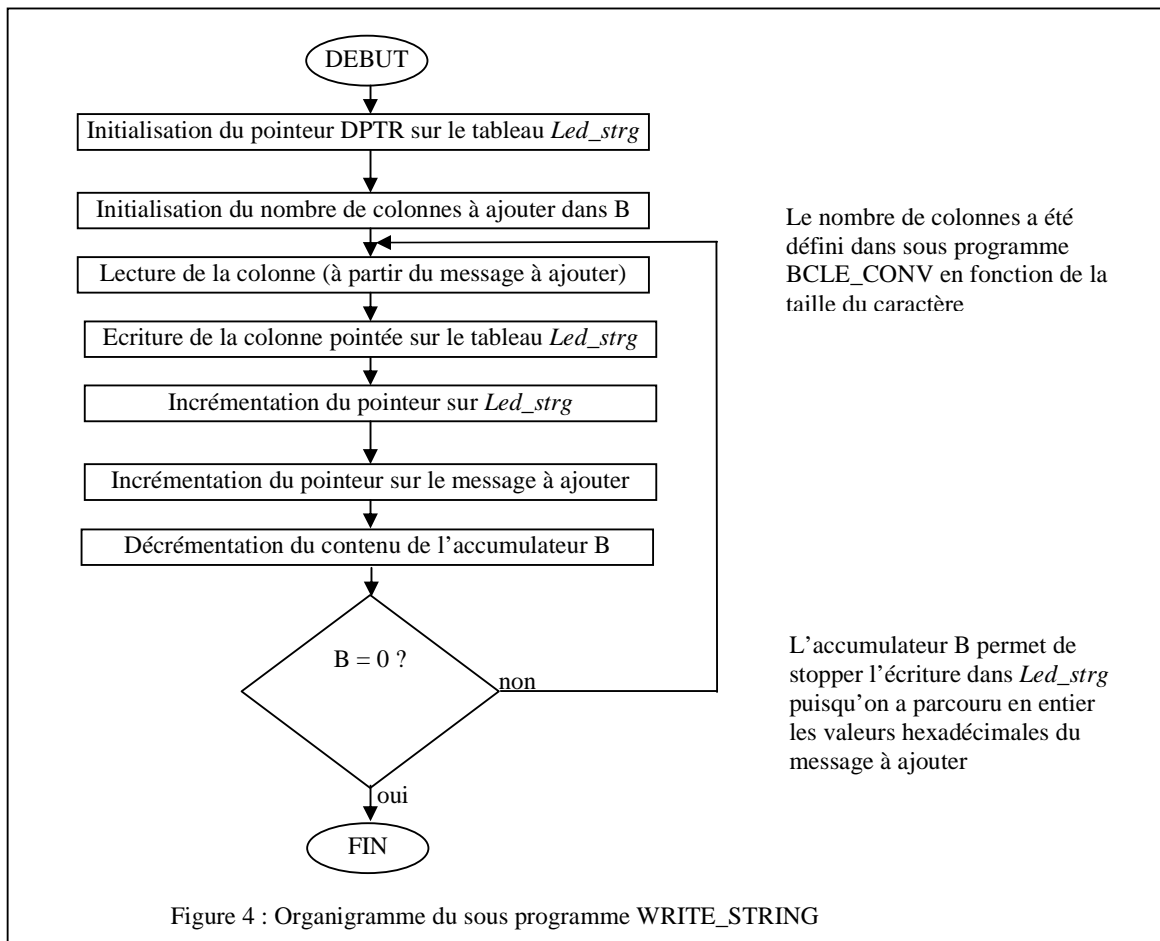


## 4. Descriptif du fichier à inclure S\_ASCII.A51

### 4.1 Sous programme CONV\_LED (Figure 5 )

Le sous programme CONV\_LED convertit caractère par caractère le tableau *Msg\_Asc* en code "DATA", mémorisé dans le tableau *Mes\_Led*. Le code "DATA" permet d'afficher les caractères alphanumériques ASCII sur le panneau de led, chaque octet correspondant à l'affichage d'une colonne. Ensuite, il détermine la valeur contenue dans l'accumulateur A (code ASCII) et fait pointer DPTR sur le tableau correspondant au caractère et ce tableau contient autant d'octet que de colonnes occupées sur le pavé de LED. Donc le sous programme effectue une comparaison caractère par caractère avec des caractères alphanumériques ou spéciaux ("'", ":", "-", "+") et s'il ne trouve pas de correspondance, on soustrait 20 (hexadécimal) au contenu de A afin de convertir en majuscule (A->Z compris entre 41 et 5A et a->z comprise entre 61 et 7A) et on recompare la valeur contenue dans A avec les codes ASCII des lettres de l'alphabet uniquement. En effet la fonction GET\_MSGE relève aussi bien les lettres minuscules que majuscules et le pavé de LED affiche seulement les lettres en majuscules.

## 4.2 Sous programme WRITE\_STRING (figure 5 )



## 5. Descriptif du fichier à inclure S\_AFILED.A51

Le fichier S\_AFILED.A51 est constitué de différentes fonctions qui permettent la gestion de l'affichage sur le panneau de led.

Une fonction INIT\_LED permet d'initialiser le panneau de led ( Figure 1 ).

La fonction AFI\_ECRAN permet l'affichage complet ligne par ligne du panneau de led à partir d'une mémoire *Mem\_Affi* contenant 16 octets successifs correspondant aux 16 colonnes (figure 2 ).

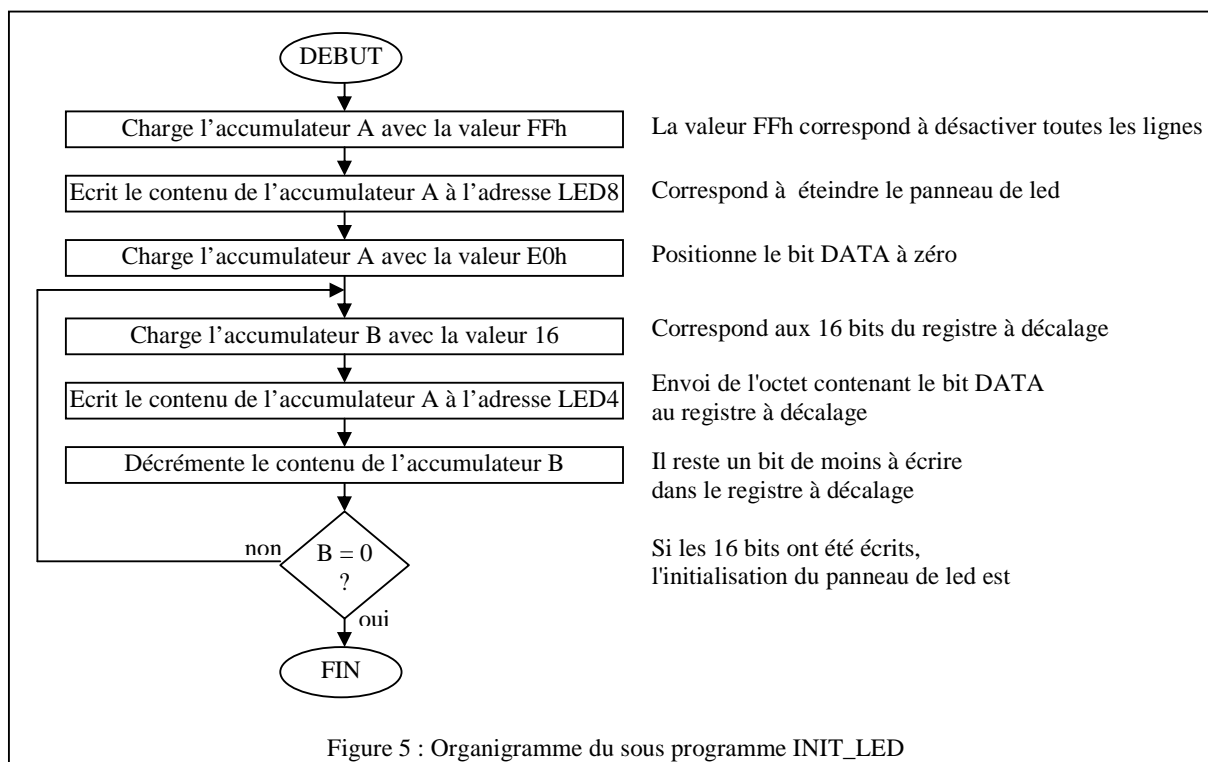
La fonction AFI\_LIGNE ( Figure 3 ) permet de gérer la configuration du registre à décalage ainsi que l'affichage d'une ligne à partir du contenu de la mémoire *Mem\_Affi* ( Figure 3 ).

La fonction AFI\_TEMPO permet de temporiser entre l'affichage de deux écrans successifs ( Figure 4 ).

La fonction DECAL\_MEM permet de modifier le contenu de la mémoire *Mem\_affi* en effectuant le décalage d'une case mémoire du contenu et en entrant un nouvel octet dans la dernière case mémoire laissée libre ( Figure 5 ).

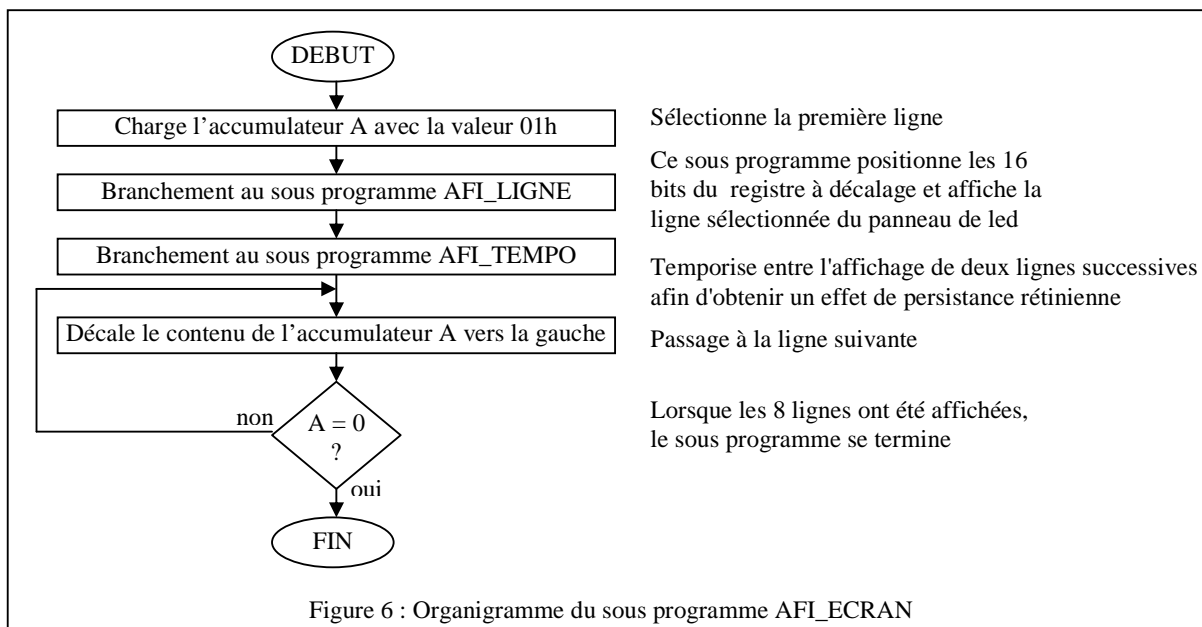
### 5.1 Sous programme INIT\_LED ( Figure 5 )

Ce sous programme réalise l'initialisation du panneau de LED en désactivant les 8 lignes, en plaçant 16 zéros dans le registre à décalage, et en validant chaque sortie du registre à décalage.



## 5.2 Sous programme AFI\_ECRAN ( Figure 6 )

Le sous programme AFI\_ECRAN gère l'affichage des 8 lignes du panneau de led en les sélectionnant une à une et en appelant la fonction d'affichage d'une ligne AFI\_LIGNE.



### 5.3 Sous programme AFI\_LIGNE (figure 7 )

Il gère l'affichage de la ligne sélectionnée à partir du contenu de la mémoire *Mem\_Affi*. Tous les bits du registre à décalage sont configurés avant de valider les sorties puis la ligne souhaitée.

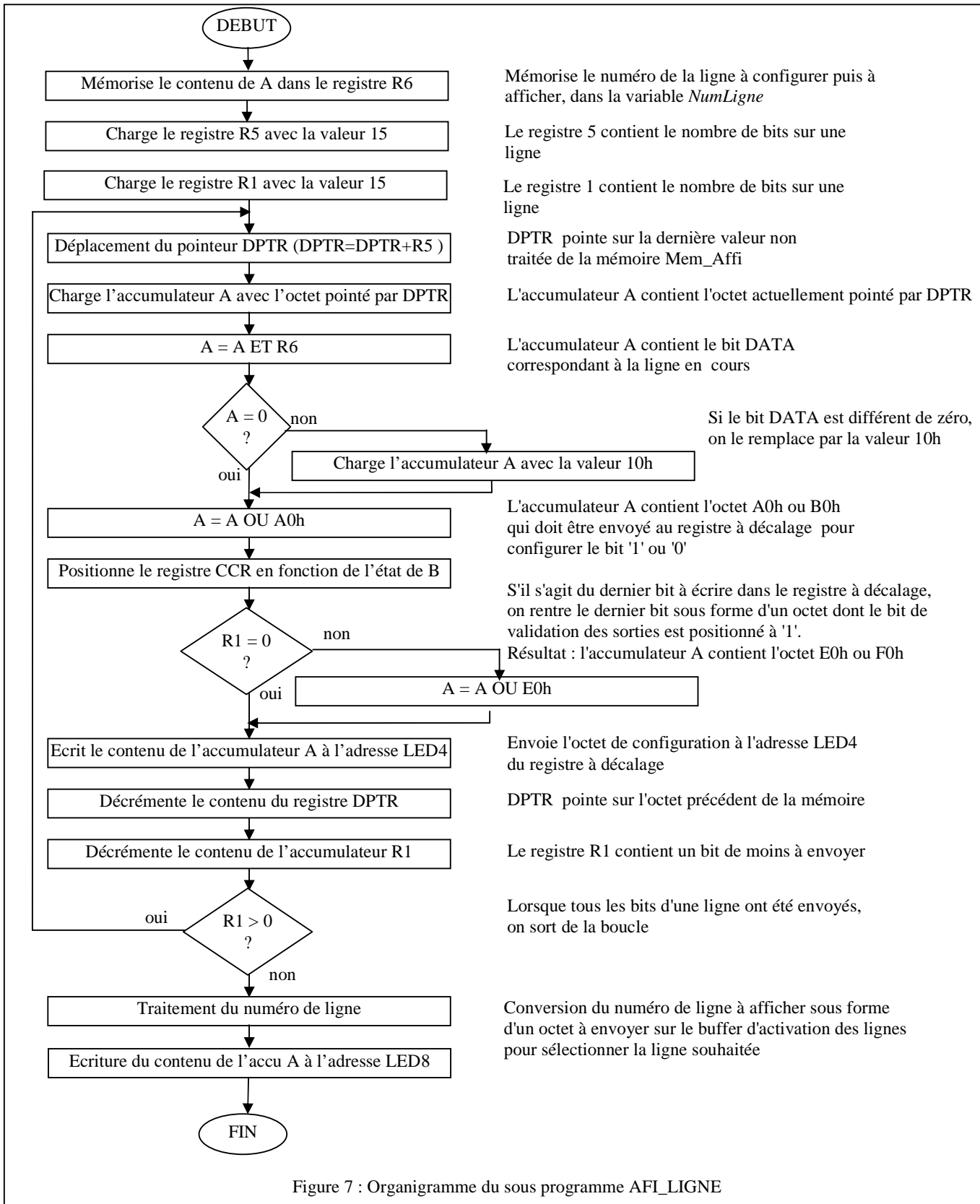


Figure 7 : Organigramme du sous programme AFI\_LIGNE

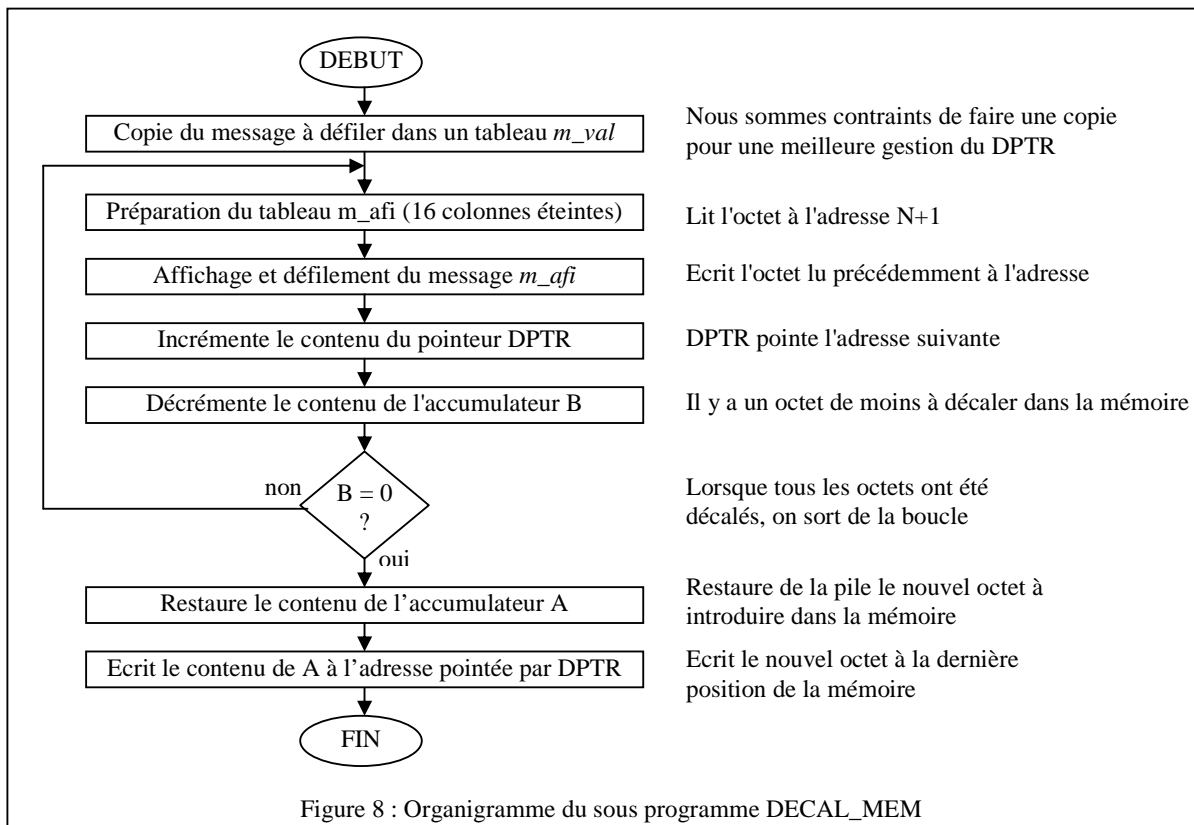
## 6. Descriptif du fichier à inclure S\_sprled.a51

Le fichier S\_sprled.a51 est constitué de différentes fonctions qui permettent le défilement de messages sur le panneau de led.

Le sous programme AFI\_TEMPO gère la temporisation entre l'affichage de deux écrans successifs.

### 6.1 Sous programme DECAL\_MEM (figure 8 )

Le sous programme DECAL\_MEM permet de faire défiler un message sur le pavé de LED. Pour cela cette fonction effectue une modification du buffer réservé à l'affichage sur le pavé de LED (16 octets car il y a 16 colonnes ). C'est à dire qu'il effectue un décalage vers la gauche des octets du buffer avec une insertion dans la colonne de droite de l'octet passé en paramètre (Accumulateur A).



## 6.2 Sous programme DEFI\_LED (figure 9)

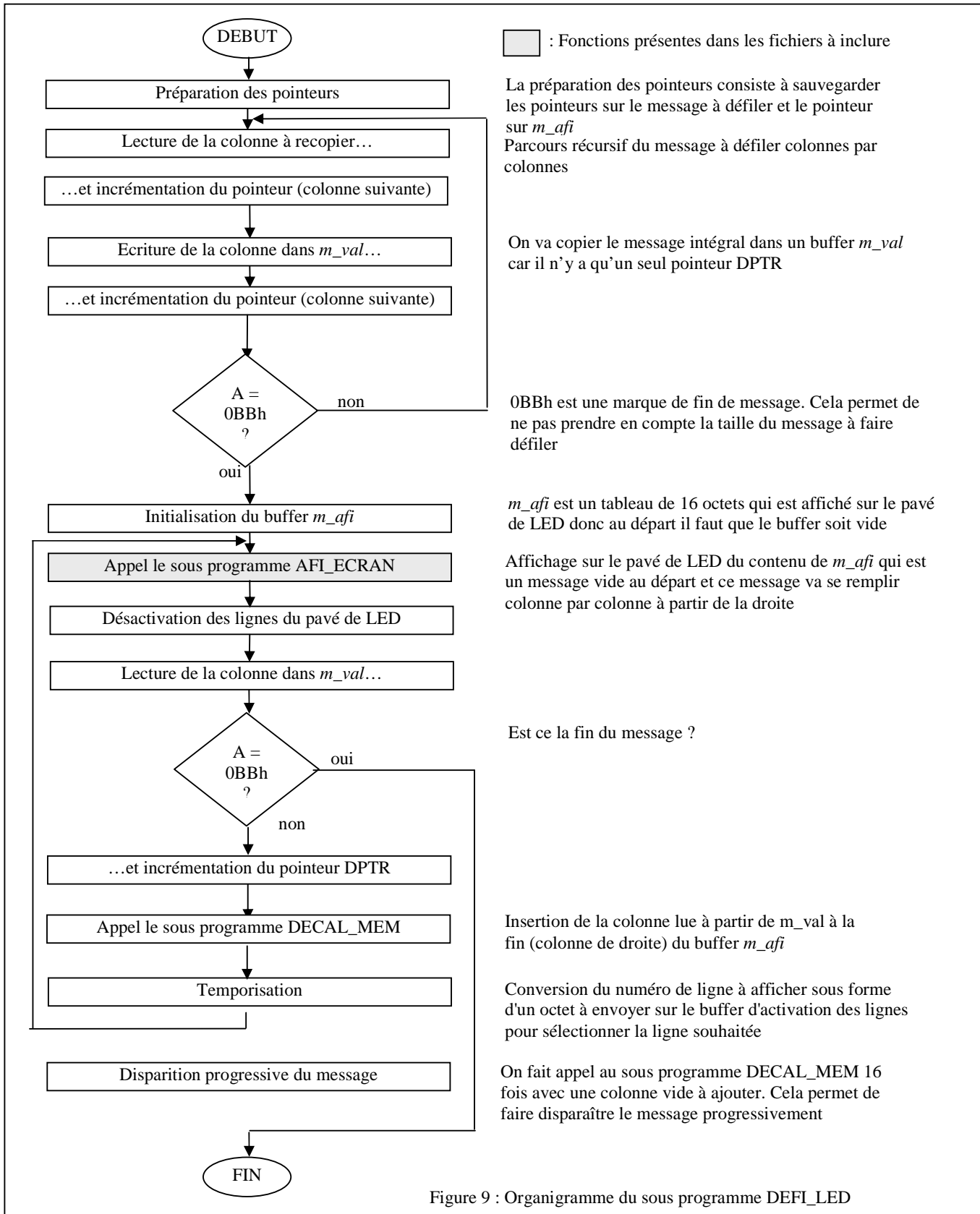
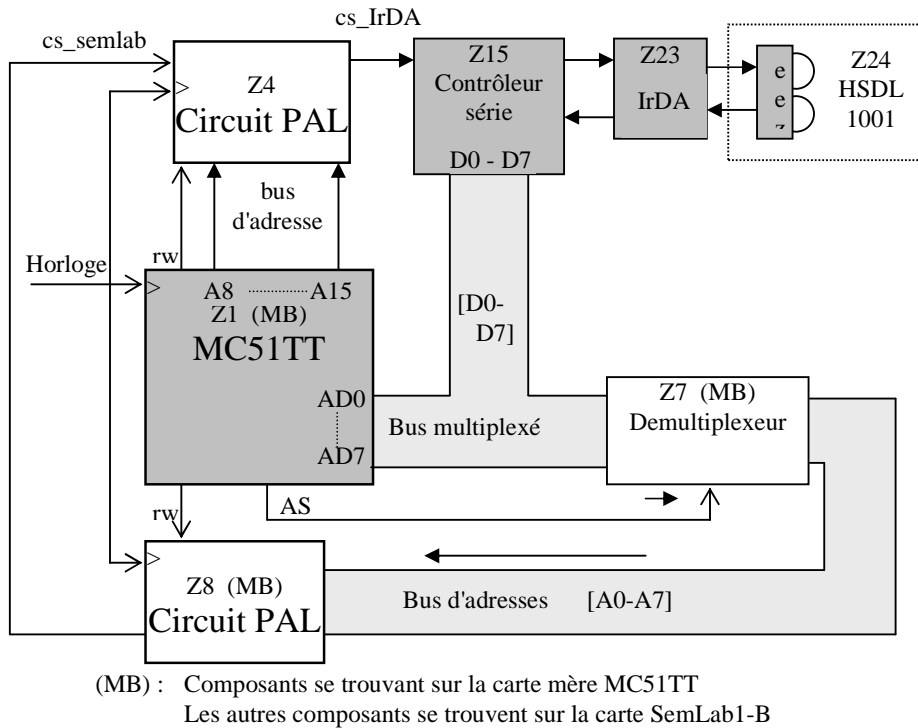


Figure 9 : Organigramme du sous programme DEFI\_LED

# VIII - COMMUNICATION IRDA

## SCC2691(Z15), HSDL7001(Z23) et HSDL1001(Z24)

### 1. Liaison microcontrôleur vers l'interface IrDA



### 2. Descriptif du fonctionnement

Le système de communication IrDA est composé de trois circuits :

- une liaison série (SCC2691),
- une interface IrDA (HSDL7001),
- un transmetteur infrarouge IrDA (HSDL1001).

Le HSDL7001 est un codeur décodeur IrDA qui sert d'interface entre les signaux série IrDA et des signaux série classique de type RS232.

Il se configure par l'envoi de l'octet 00h à l'adresse 801Ah.

L'UART SCC2691 est composé de 11 registres accessibles par les adresses comprise entre 8010h et 8017h (voir la documentation constructeur).

Voir les fichiers P\_IrDAChif.A51 et P\_IrDACharIT.A51

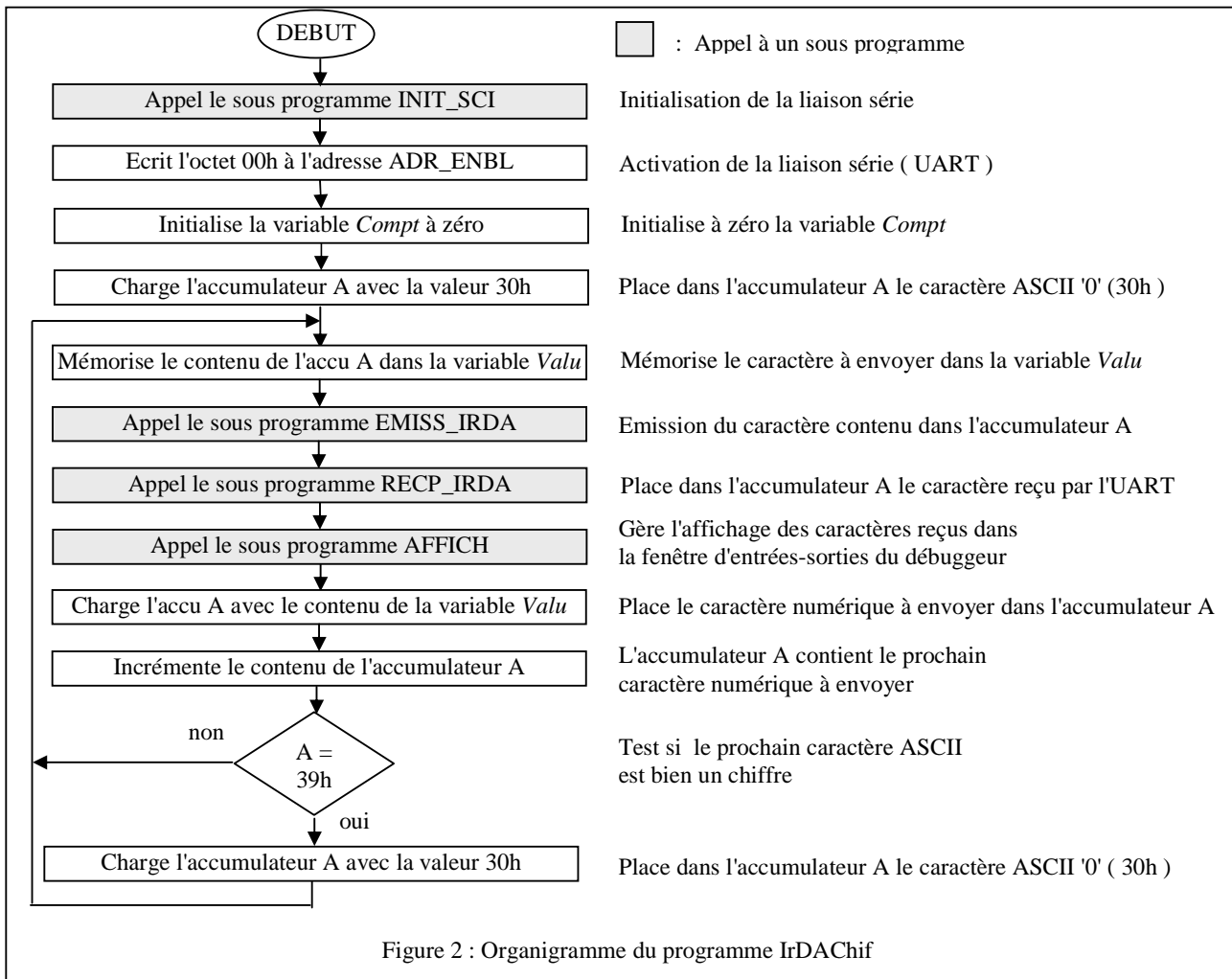
### 3. Descriptif du programme P\_IrDACHif.A51

Le programme P\_IrDACHif permet d'afficher suite numérique allant de 0 jusqu'à 9 dans la fenêtre d'entrées-sorties du débogueur via la liaison IrDA.

Ce programme utilise pour cela une routine qui est intégrée au moniteur. La routine Tx\_OCTET affiche dans la fenêtre d'entrées-sorties du débogueur le caractère ASCII contenu dans l'accumulateur A.

Ce programme utilise le principe de synchronisation suivant. La liaison série émet un caractère puis attend de le recevoir par lecture du registre d'état avant de l'afficher si le caractère reçu correspond au caractère envoyé.

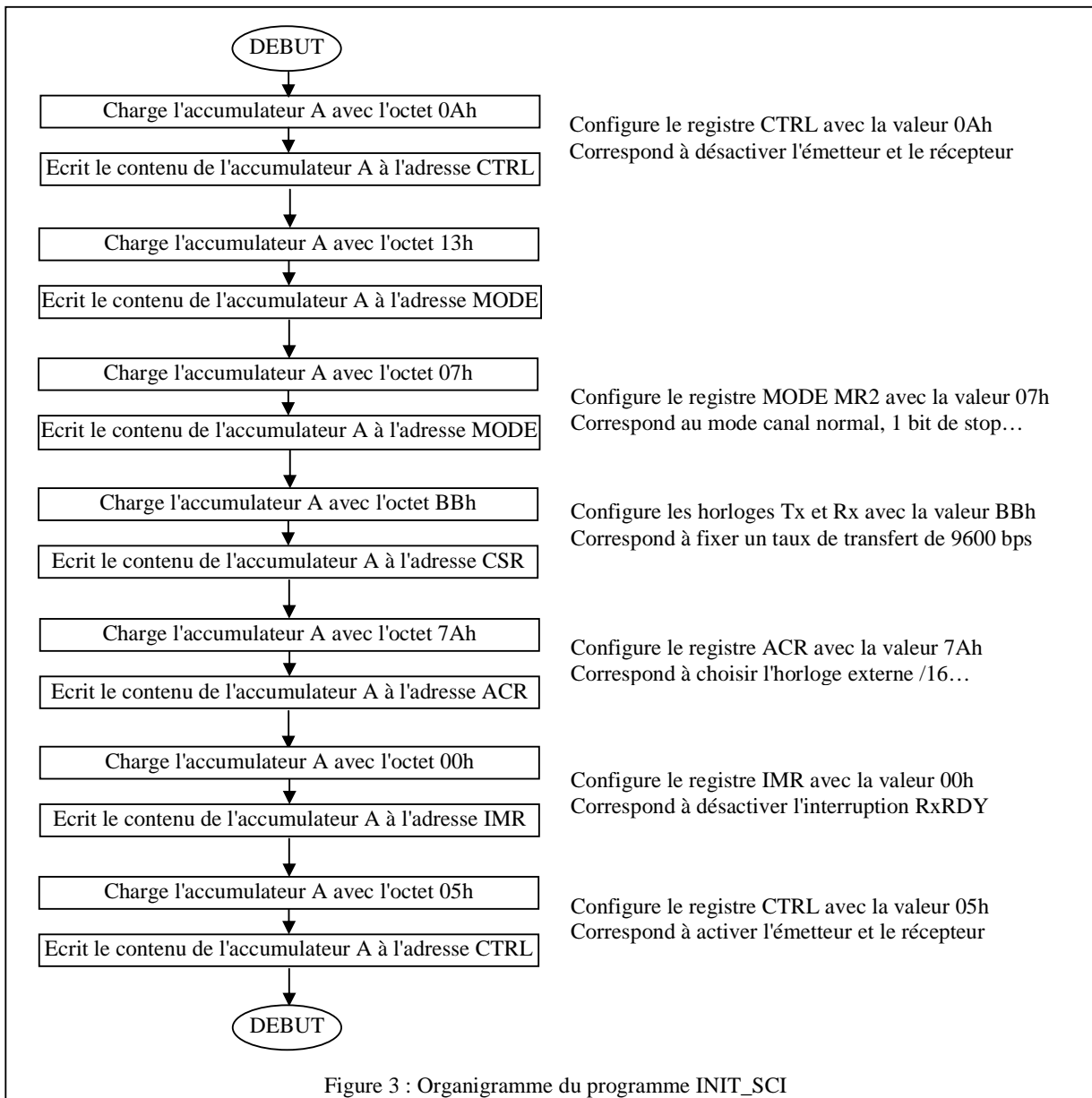
#### 3.1 Organigramme du programme P\_IrDACHif.A51 (Figure 1)



### 3.2 Sous programme INIT\_SCI

( Figure 3 )

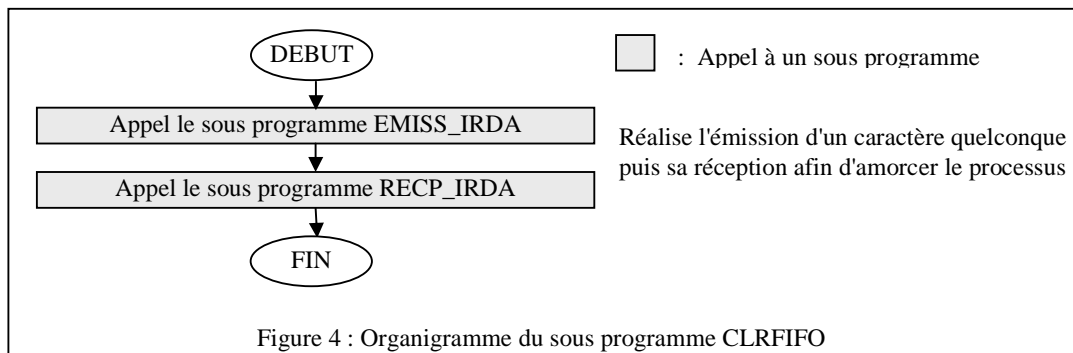
Le sous programme INIT\_SCI permet de configurer les modes de fonctionnement de la liaison série et d'initialiser ses différents registres.



### 3.3 Sous programme CLR\_FIFO

( Figure 4 )

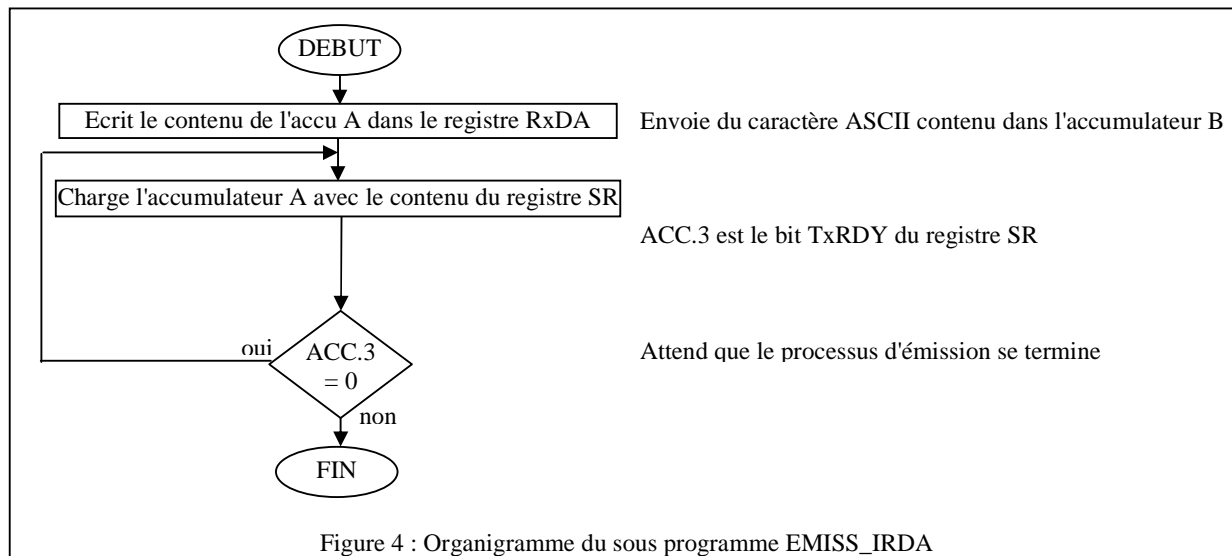
Le programme CLR\_FIFO a pour fonction d'amorcer le processus d'émission et de réception de la liaison série. Les caractères émis et reçu sont quelconques.



### 3.4 Sous programme EMISS\_IRDA

( Figure 4 )

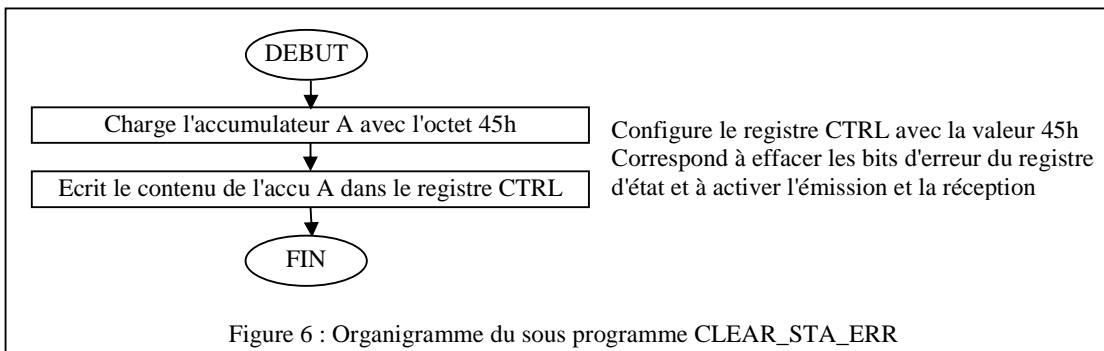
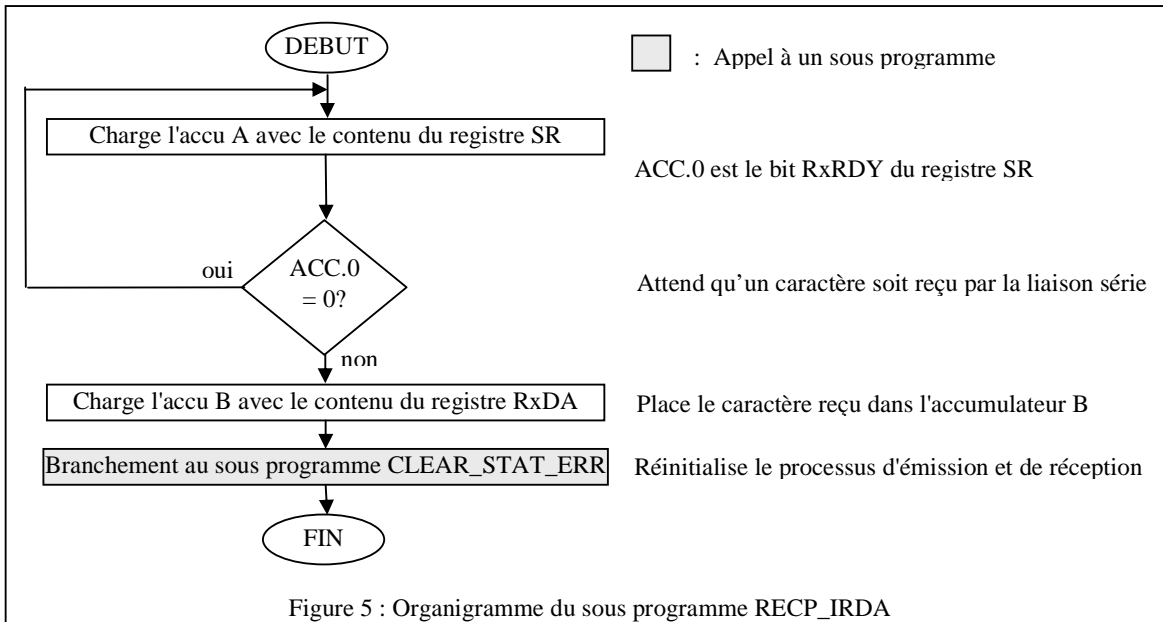
Le sous programme EMISS\_IRDA émet par le biais de la liaison série le caractère contenu dans l'accumulateur A puis attend que le processus d'émission se finisse avant de se terminer.

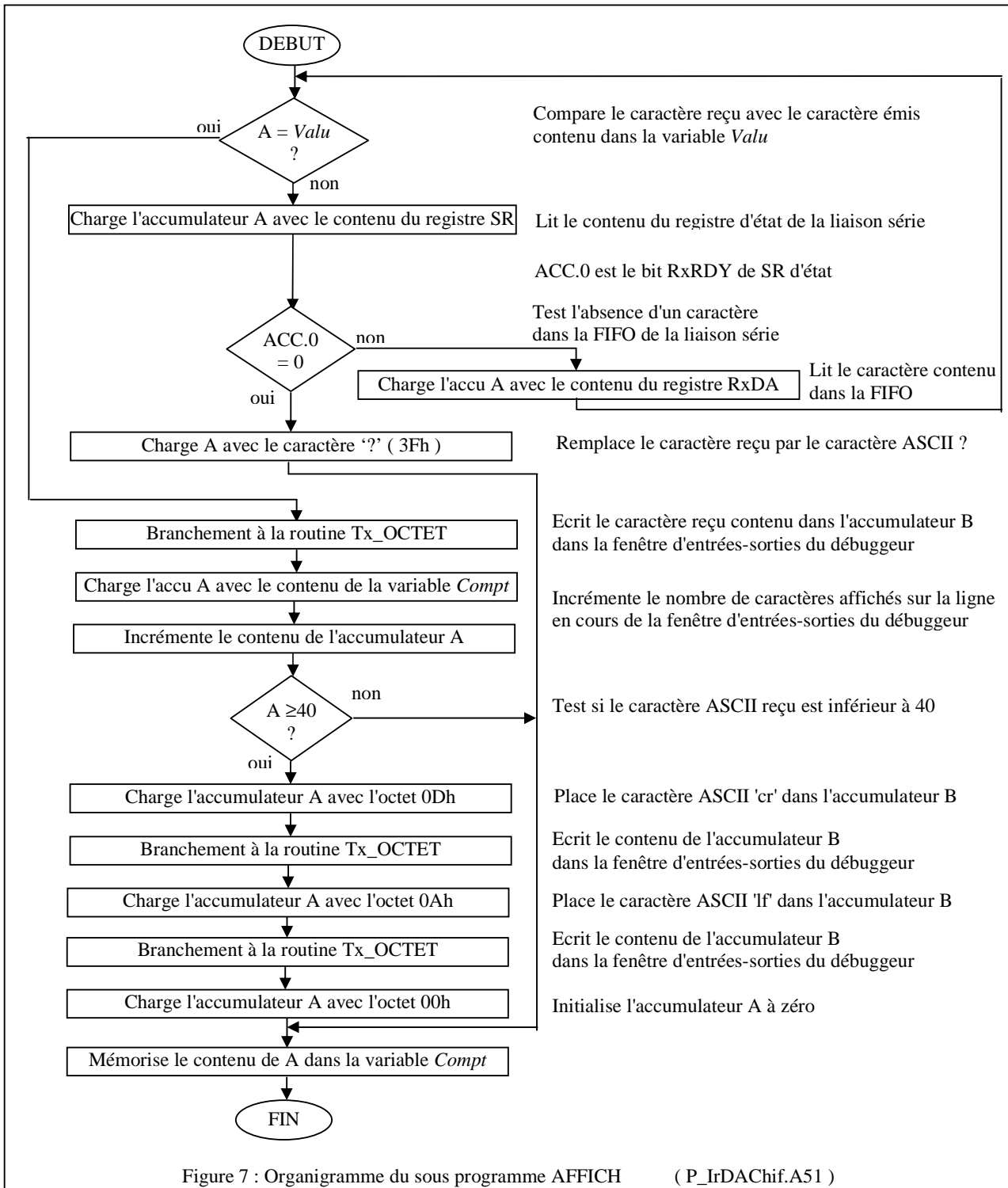


### 3.5 Sous programme RECP\_IRDA

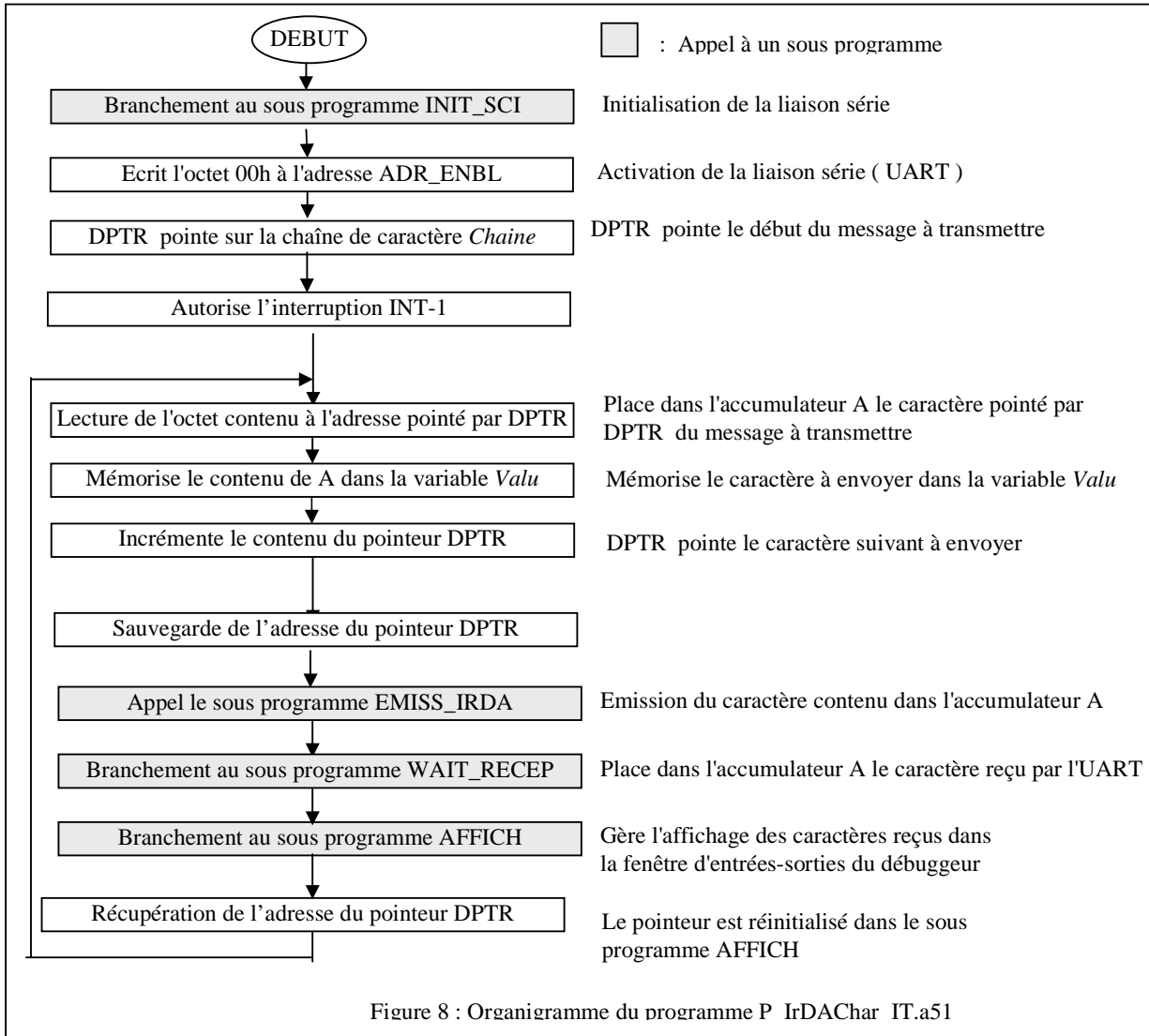
( Figure 5 )

Ce sous programme lit le registre d'état de la liaison série pour être informée lors de la réception d'un caractère puis place ce dernier dans l'accumulateur A avant de lancer le sous programme CLEAR\_STAT\_ERR.



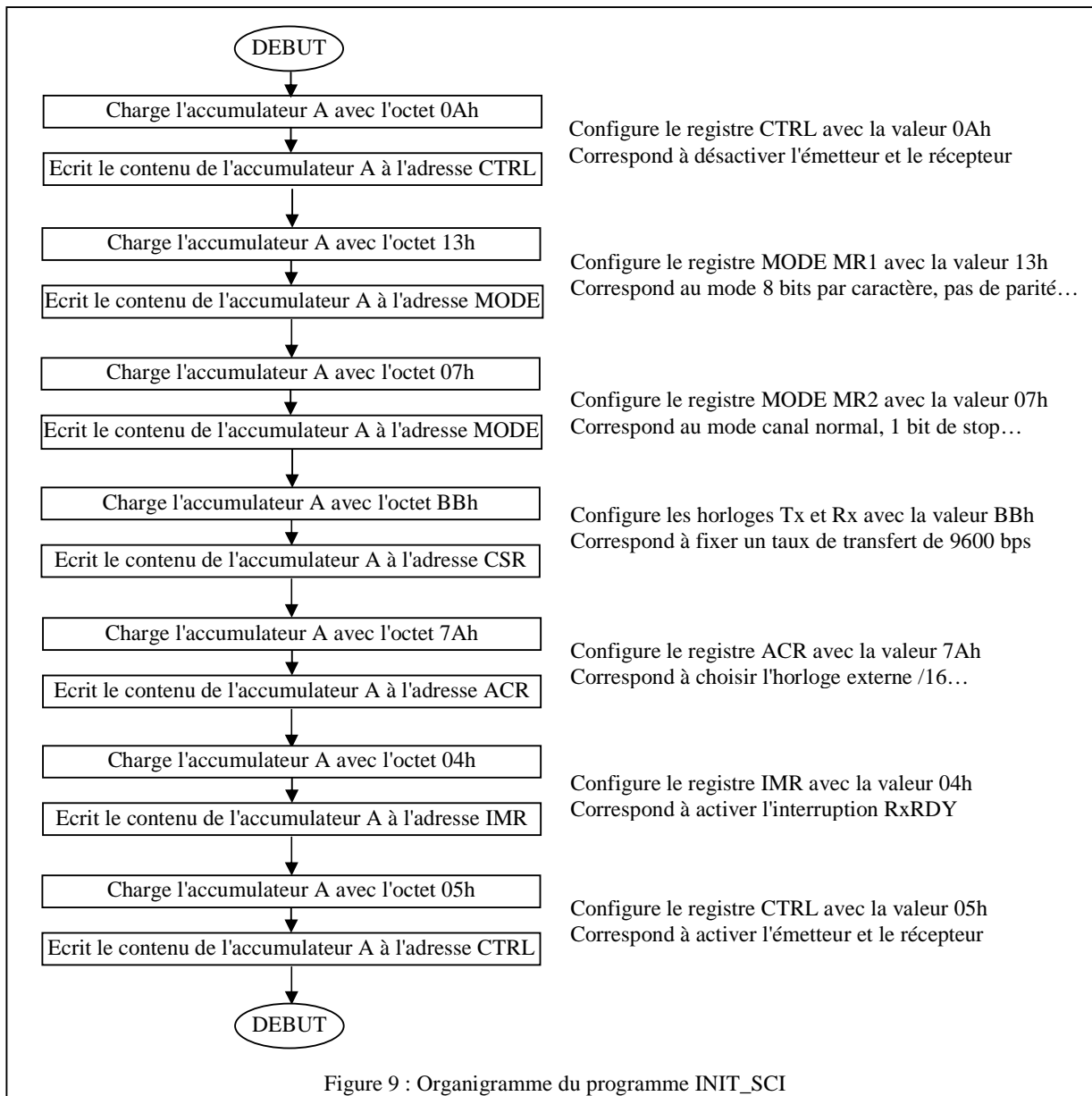


A la différence du programme précédent, celui-ci utilise le principe des interruptions. Le programme émet un caractère sur la liaison série puis attend que la survenue d'une interruption déclenchée par la réception d'un caractère lui soit signalée par un flag (drapeau). Lorsque cela se produit, le caractère reçu est affiché dans la fenêtre d'entrées-sorties du débogueur s'il correspond au caractère émis.



## 4.2 Sous programme INIT\_SCI

(figure 9 )

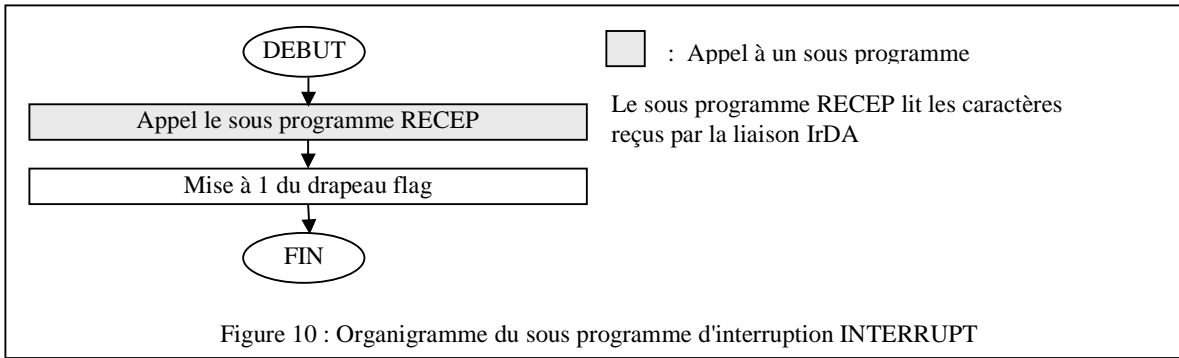


## 4.3 Sous programme INTERRUPT

(figure 10 )

Le sous programme INTERRUPT est un sous programme d'interruption qui est exécuté lors de la survenue d'une interruption externe sur la broche adéquate du microcontrôleur. Cela est rendu possible par l'écriture de l'adresse de ce sous programme à l'adresse 0013h.

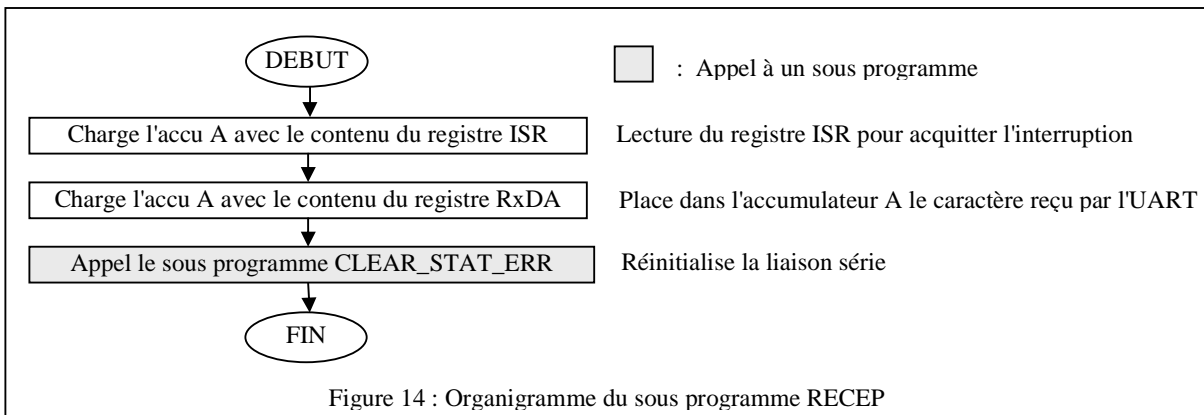
Ce sous programme lance donc le sous programme de réception de caractère puis incrémente la variable flag\_recept avant de se terminer.



#### 4.4 Sous programme RECEP

(figure 11 )

Ce sous programme acquitte l'interruption avant de lire le caractère reçu et de réinitialiser le processus d'émission et de réception.



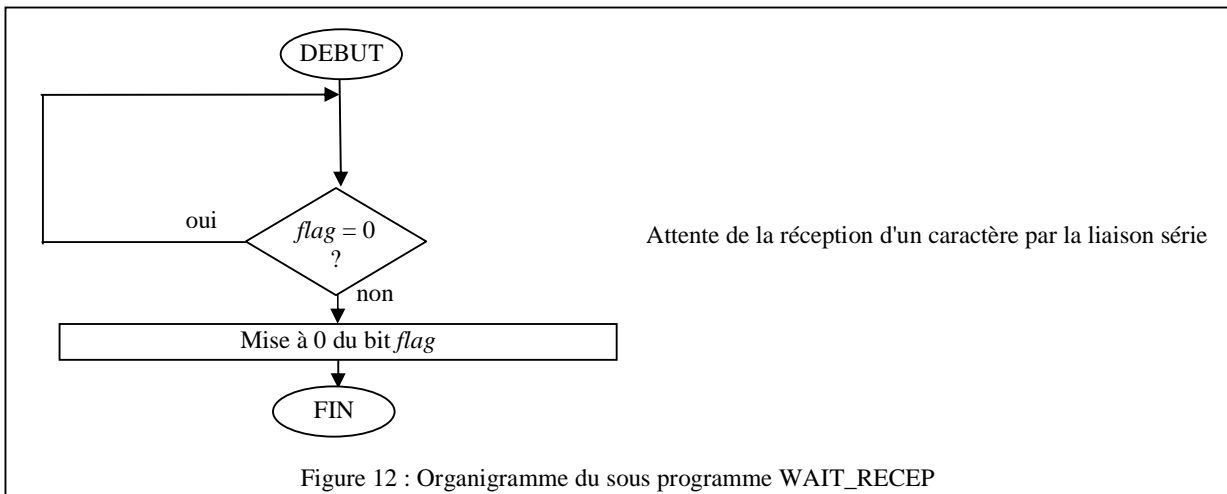
#### 4.5 Sous programme EMISS\_IRDA

Cette fonction est présentée dans le paragraphe 3.4

#### 4.6 Sous programme WAIT\_RECEP (figure 12 )

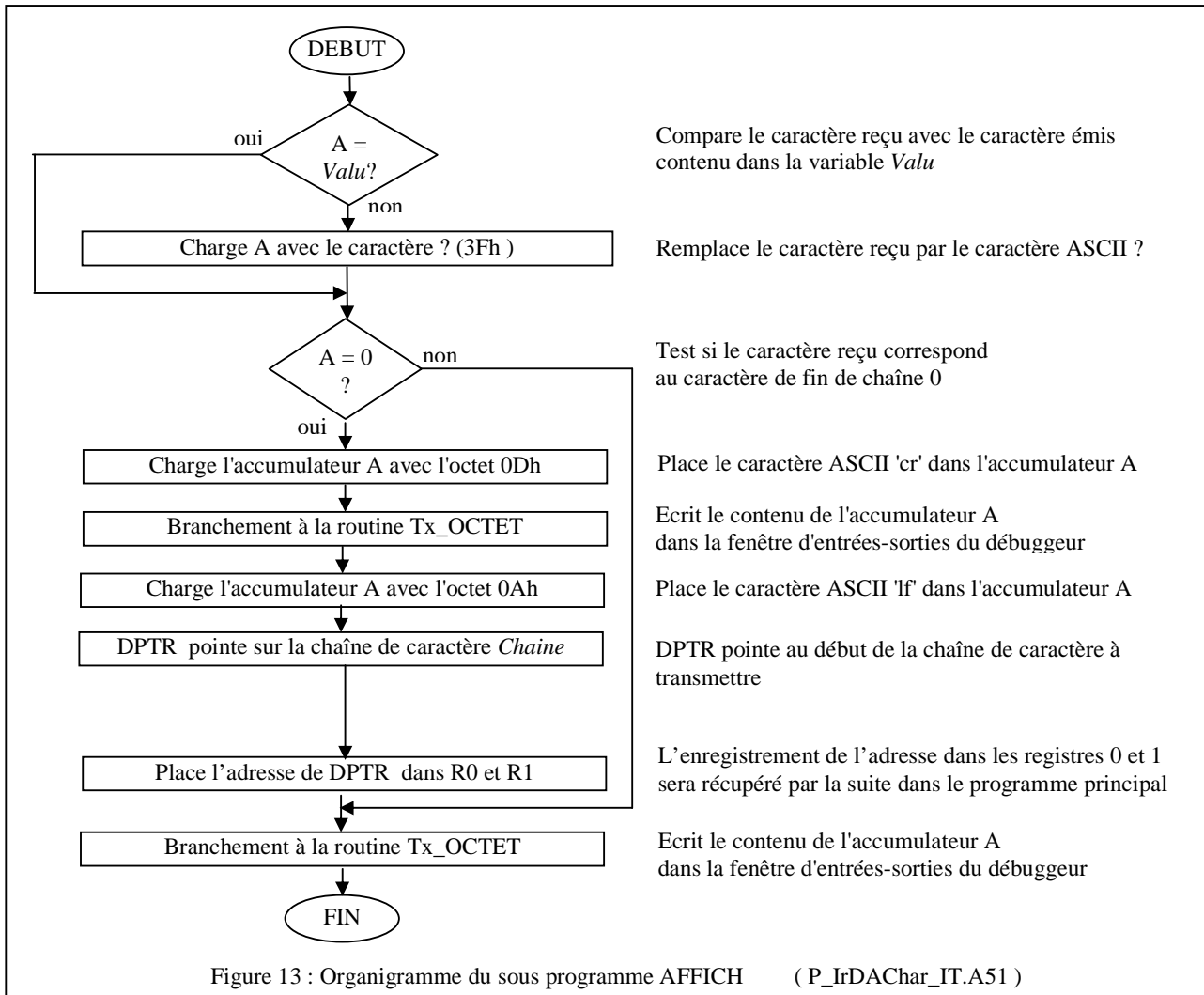
Ce sous programme attend que le bit *flag* passe à 1 signalant ainsi qu'une interruption, provoquée par la réception d'un caractère sur la liaison série s'est produite.

La variable *flag* est alors réinitialisée à zéro et le programme peut alors se poursuivre par l'exécution du sous programme AFFICH.

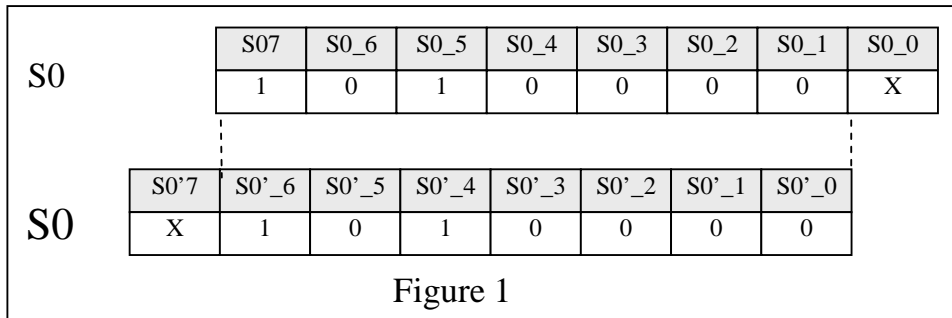


4.7 Sous programme AFFICH (P\_IrDACHar\_IT.A51) (figure 13 )

Le sous programme AFFICH place dans l'accumulateur A le caractère reçu avant de l'afficher dans la fenêtre d'entrées-sorties du débogueur s'il correspond au caractère émis.







L'accès au registre S0' se fait par le bus de données à l'adresse 8002h (A0 = 0), après que le registre S1 ait préalablement été correctement configuré (l'écriture du registre S1 se fait à l'adresse 8003h). La valeur par défaut de l'adresse contenue dans le registre S0' est 00h.

### Registre *CLOCK* S2

Le registre S2 permet de régler la fréquence d'horloge interne du circuit Fclk ainsi que la fréquence d'Horloge Fsc1 du bus I<sup>2</sup>C.

S2	0	0	0	S2_4	S2_3	S2_2	S2_1	S2_0
----	---	---	---	------	------	------	------	------

La fréquence d'horloge Fsc1 est réglable à partir des bits S2\_1 et S2\_0 (tableau 1).

**Tableau 1**      Registre S2 sélection de la fréquence SCL

BIT		FREQUENCE FCL APPROXIMATIVE Fsc1 (kHz)
S2_1	S2_0	
0	0	90
0	1	45
1	0	11
1	1	1.5

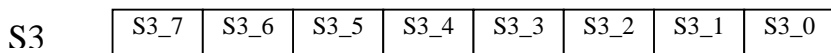
La fréquence Fclk doit correspondre, par le biais des bits S2\_4, S2\_3 et S2\_2, à celle fournie par le microcontrôleur sur la broche d'entrée du même nom (tableau 2).

**Tableau 2**      Registre S2 sélection de la fréquence d'horloge

FREQUENCE D'HORLOGE INTERNE			
S2_4	S2_3	S2_2	Fclk (MHz)
0	X	X	3
1	0	0	4.43
1	0	1	6
1	1	0	8
1	1	1	12

### Registre *INTERRUPT VECTOR* S3

Le registre *interrupt vector* fournit un vecteur d'interruption 8 bits programmable par l'utilisateur pour les interruptions vectorisées du microcontrôleur.

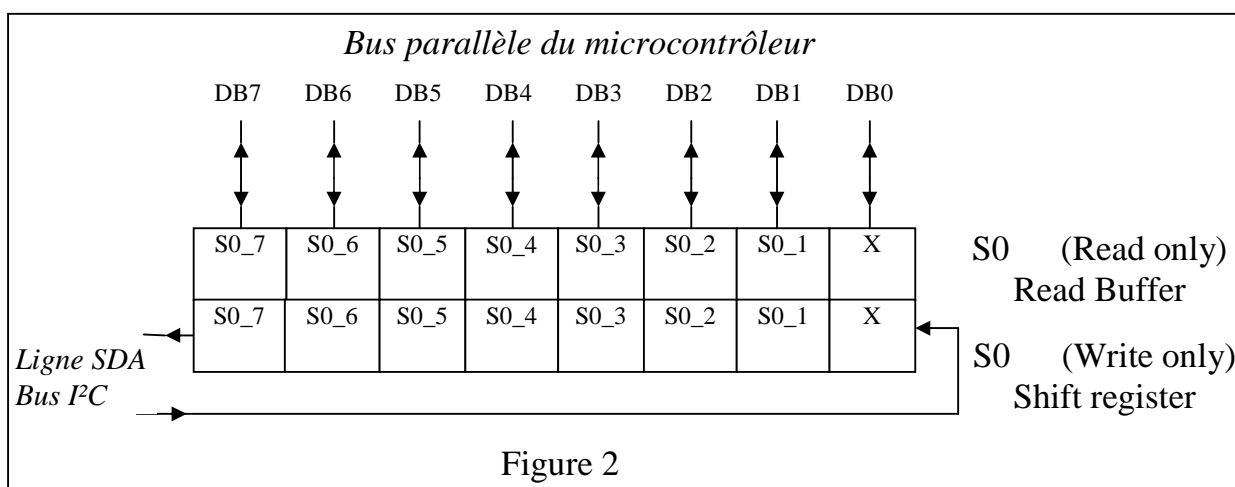


Le vecteur est envoyé sur le bus parallèle (DB7 à DB0) lorsqu'un signal *interrupt acknowledge* est reçu et que le flag ENI (Enable Interrupt) du registre de contrôle S1 est positionné à '1'.

La valeur par défaut du registre, au reset est 0Fh.

### Registre *DATA SHIFT/READ BUFFER* S0

Le registre S0 agit comme un registre à décalage série et un buffer de lecture interfacé au bus I<sup>2</sup>C. Tous les échanges de données que ce soit en lecture ou en écriture se font à partir de ce registre. S0 est une combinaison d'un registre à décalage et d'un buffer de données ; les données parallèles sont toujours écrites dans le registre à décalage et lues dans le buffer de données. Les données du bus I<sup>2</sup>C sont toujours émises ou reçues à partir du registre à décalage S0 (Figure 2).



### Registre *CONTROL/STATUS* S1

Le registre S1 contrôle les opérations du bus I<sup>2</sup>C et fournit des informations sur l'état de celui-ci.

L'accès au registre S1 se fait à l'adresse 8003h (A0 = 1).

Pour une communication plus efficace entre le microcontrôleur et le bus I<sup>2</sup>C, chacun des bits des fonctions de lecture et d'écriture du registre S1 sont séparés (Tableau 3).

**Tableau 3** Registre de contrôle et d'état S1

CONTROL/STATUS	BITS								MODE
	PIN	ES0	ES1	ES2	ENI	STA	STO	ACK	
Control	PIN	ES0	ES1	ES2	ENI	STA	STO	ACK	Write only
Status	PIN	0	STS	BER	AD0/LRB	AAS	LAB	!BB	Read only

1) Registre de contrôle S1

PIN :

Le bit PIN lorsqu'il est positionné à '1' permet d'initialiser à '0' les bits du registre d'état, réalisant ainsi un reset logiciel.

ES0 :

ES0 permet d'activer ou de désactiver les entrées / sorties du bus série I<sup>2</sup>C. Lorsque ce bit est à '0', l'accès au registre pour son initialisation est possible. Lorsque le bit ES0 est à '1', les communications du bus I<sup>2</sup>C sont activées.

ES1 et ES2 :

Les bits ES1 et ES2 contrôlent la sélection des autres registres pour l'initialisation et le contrôle des opérations standard. Après que ces bits aient été programmés pour accéder au registre désiré (Tableau 4), le registre est sélectionné par un accès à l'adresse 8002h.

**Tableau 4** Accès aux registres du PCF8584

Adressage des registres internes				
A0	ES1	ES2	!IACK	FONCTION
ES0 = 0 ; interface série désactivée				
1	0	X	1*	R/W S1 : control
0	0	0	1*	R/W S0' : (own adress)
0	0	1	1*	R/W S3 : (interrupt vector)
0	1	0	1*	R/W S2 : (clock register)
ES0 = 1 ; interface série activée				
1	0	X	1	W S1 : control
1	0	X	1	R S1 : status
0	0	0	1	R/W S0 : (data)
0	0	1	1	R/W S3 : (interrupt vector)
X	0	X	0	R S3 : (interrupt vector ACK cycle)

\* : 'X' si ENI = 0.

ENI :

Ce bit active la sortie d'interruption externe ! INT, qui est généré lorsque le bit PIN est actif (niveau bas).

STA et STO :

Ces bits contrôlent la génération de la condition START du bus I<sup>2</sup>C avec transmission de l'adresse esclave du PCF8584, ainsi que la génération de la condition STOP.

**Tableau 5** Instructions de contrôle du bus série

STA	STO	FONCTION
0	0	NOP *
0	1	STOP
1	0	START
1	1	DATA CHAINING

\* : **NOP est un mode non opérationnel**

ACK :

Ce bit est normalement positionné à '1'. Cela contraint le contrôleur de bus I<sup>2</sup>C à envoyer un 'acknowledge' automatiquement après chaque émission d'un octet. Il doit être positionné à '0' lorsque le contrôleur de bus I<sup>2</sup>C opère en mode maître / réception et ne requière pas d'autre envoi de données de la part de l'esclave.

Registre d'état S1

PIN (Pending Interrupt Not):

- Le bit PIN peut être utilisé dans les applications de polling pour tester la fin d'une transmission série. Lorsque le bit ENI est également positionné à '1', le flag PIN déclenche l'interruption externe via la sortie /INT.

L'activation du bit STA (bit START) positionne le bit PIN à '1' (inactif).

Dans le mode émetteur, après une transmission réussie d'un octet sur le bus I<sup>2</sup>C, le bit PIN est automatiquement repositionné à '0' (ACTIF) indiquant la fin de transmission de l'octet.

Dans le mode émetteur, le bit PIN est positionné à '1' (inactif) à chaque accès au registre S0 en écriture.

Dans le mode récepteur, le bit PIN est positionné à '0' (actif) à la fin de la réception de chaque octet. La ligne SCL sera maintenue au niveau bas jusqu'à ce que le bit PIN soit positionné à '1'.

Dans le mode récepteur, lorsque le registre S0 est lu, le bit PIN est positionné au niveau logique '1' (inactif).

Dans le mode récepteur esclave, une condition STOP du bus I<sup>2</sup>C entraîne la mise à '0' du bit PIN (actif).

Le bit PIN est positionné à '0' lors de l'occurrence d'une condition 'bus error' (BER).

STS : non utilisé

BER (Bus Error) :

Une condition STOP ou START mal placée à été détectée. Positionne le bit /BB à '0' (inactif), et le bit PIN à '1' (actif).

LRB / AD0 (Last Received Bit or Adress 0 -General Call- ):

Ce bit qui à deux fonctions n'est valide que lorsque le bit PIN est positionné à '0'.

LRB garde la valeur du dernier bit reçu sur le bus I<sup>2</sup>C tant que le bit AAS est positionné à '0' (non adressé comme un esclave). Normalement, ce bit correspond au 'acknowledge' de l'esclave. On peut donc tester si le 'acknowledge' de l'esclave a été réalisé à travers ce bit.

AD0 ; Lorsque AAS est positionné à '1' (condition Adressed As Slave), cela signifie que le contrôleur de bus I<sup>2</sup>C à été adressé en esclave. Sous cette condition, ce bit devient le bit 'AD0' et passera à au niveau logique '1' si l'adresse esclave reçue est l'adresse 'Général Call' : 00h, ou au niveau logique '0' s'il s'agissait de l'adresse esclave du contrôleur de bus I<sup>2</sup>C.

AAS (Adressed As Slave):

Ce bit est valide uniquement lorsque le bit 'PIN' est positionné à '0'. Lorsqu'il est utilisé en récepteur esclave, ce flag est positionné à '1' lorsqu'une adresse qui arrive sur le bus I<sup>2</sup>C correspond à la valeur du registre *OWN ADDRESS S0* (décalée d'un bit, voir Figure 1), ou bien si l'adresse 'Général Call' du bus I<sup>2</sup>C (00h) a été reçue (lors d'un 'Général Call', le bit AD0 est également positionné à '1').

LAB (Lost Arbitration Bit) non utilisé

/ BB (Bus Busy):

C'est un flag en lecture seule qui indique lorsque le bus I<sup>2</sup>C est utilisé. Un zéro indique que le bus est occupé, et que son accès est impossible. Ce bit est positionné à '1' / '0' par la condition STOP / START.

### 3. Descriptif de la programmation du PCF8584

Une écriture à l'adresse 8003h (A0 = 1) programme le registre de contrôle du circuit.  
Une lecture à l'adresse 8003h (A0 = 1) retourne la valeur courante du registre d'état.

Une écriture à l'adresse 8002h (A0 = 0) entraîne l'écriture d'un octet par le circuit émetteur sur le bus I<sup>2</sup>C.  
Une lecture à l'adresse 8002h (A0 = 0) retourne la valeur du dernier octet reçu sur le bus I<sup>2</sup>C.

Lorsque le circuit est programmé en maître et qu'une donnée est écrite à l'adresse 8002h, cette donnée est envoyée sur le bus I<sup>2</sup>C. A la fin du transfert, une interruption est demandée au microcontrôleur (si le bit du registre de contrôle correspondant est validé). Cette interruption est reliée à INT-0 pour le MC51TT.

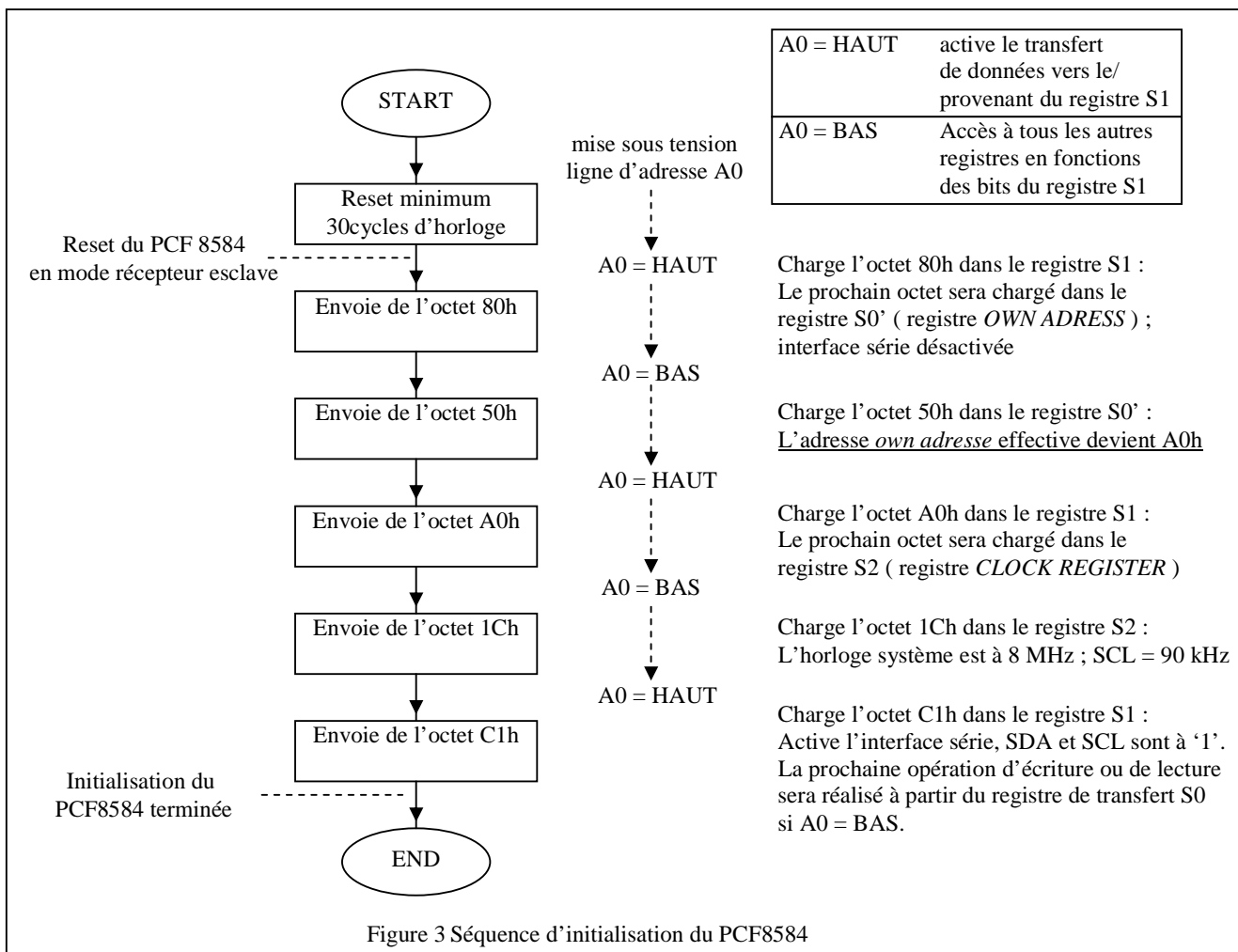
Lorsque le circuit est programmé en esclave et qu'il reçoit un octet sur le bus I<sup>2</sup>C, une interruption est envoyée (si le bit du registre de contrôle correspondant est validé) au microcontrôleur.

Si on n'utilise pas les interruptions, on peut savoir si une émission ou une réception est terminée en testant le registre d'état.

En émission (mode maître), on peut programmer le registre de contrôle pour retourner le circuit en réception (mode esclave) à la fin de l'émission.

En réception, on peut programmer le registre de contrôle pour sortir ou non un bit ACKNOWLEDGE à la fin de la réception. Quand l'un des circuits PCF 8574, PCF 8591 ou DS1621 émet un octet vers le PCF 8584, il attend en retour un bit ACKNOWLEDGE. Tant qu'il le recevra, il émettra un autre octet, par conséquent la transmission s'arrêtera quand il n'en recevra plus.

Le système utilise trois fonctions pour communiquer sur le bus I<sup>2</sup>C, qui se trouvent dans le fichier S\_I2CP84.A51 et qui sont : la fonction d'initialisation du bus I<sup>2</sup>C (Figure 3), d'émission (Figure 4) et de réception de donné (Figure 5).



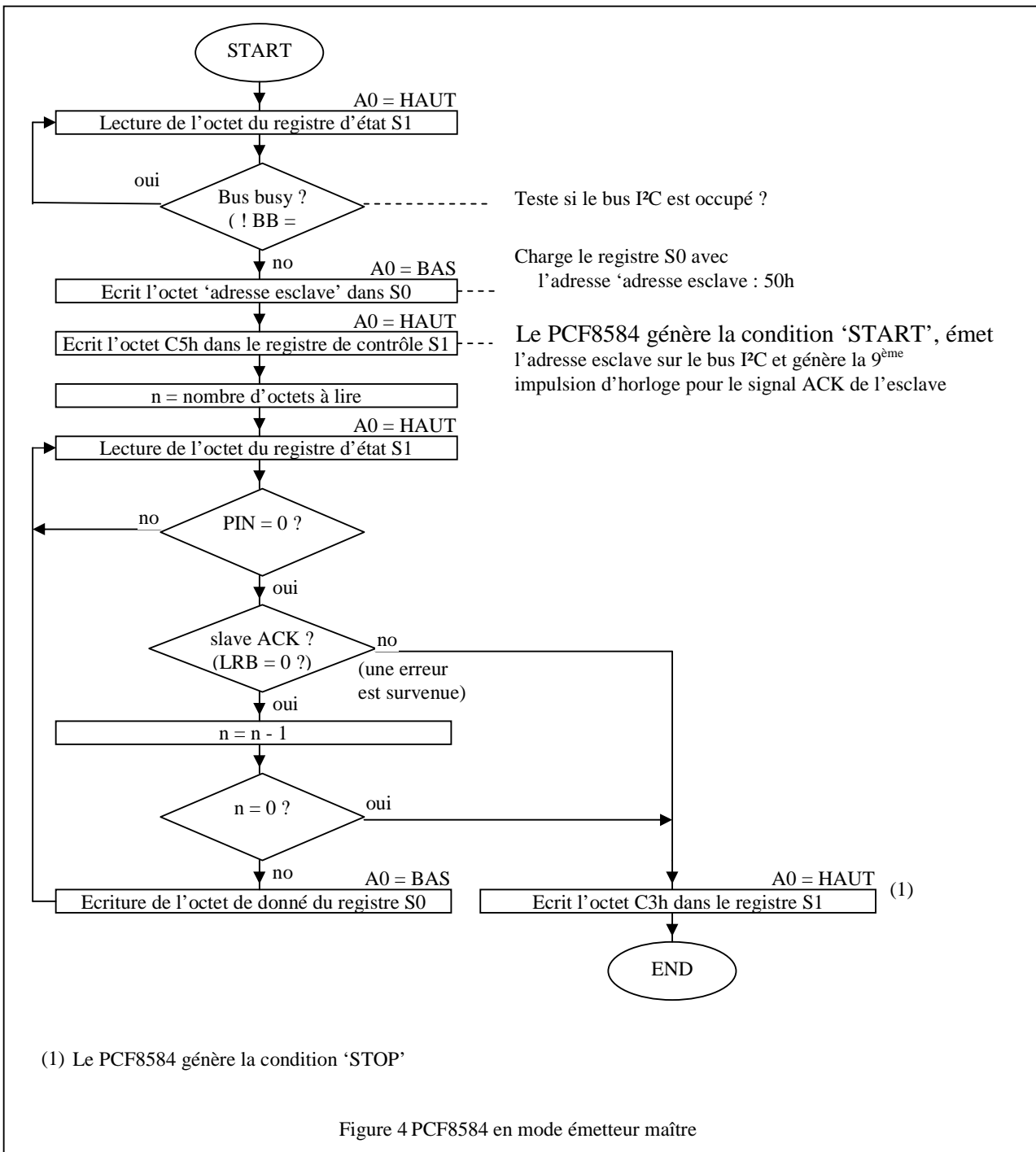
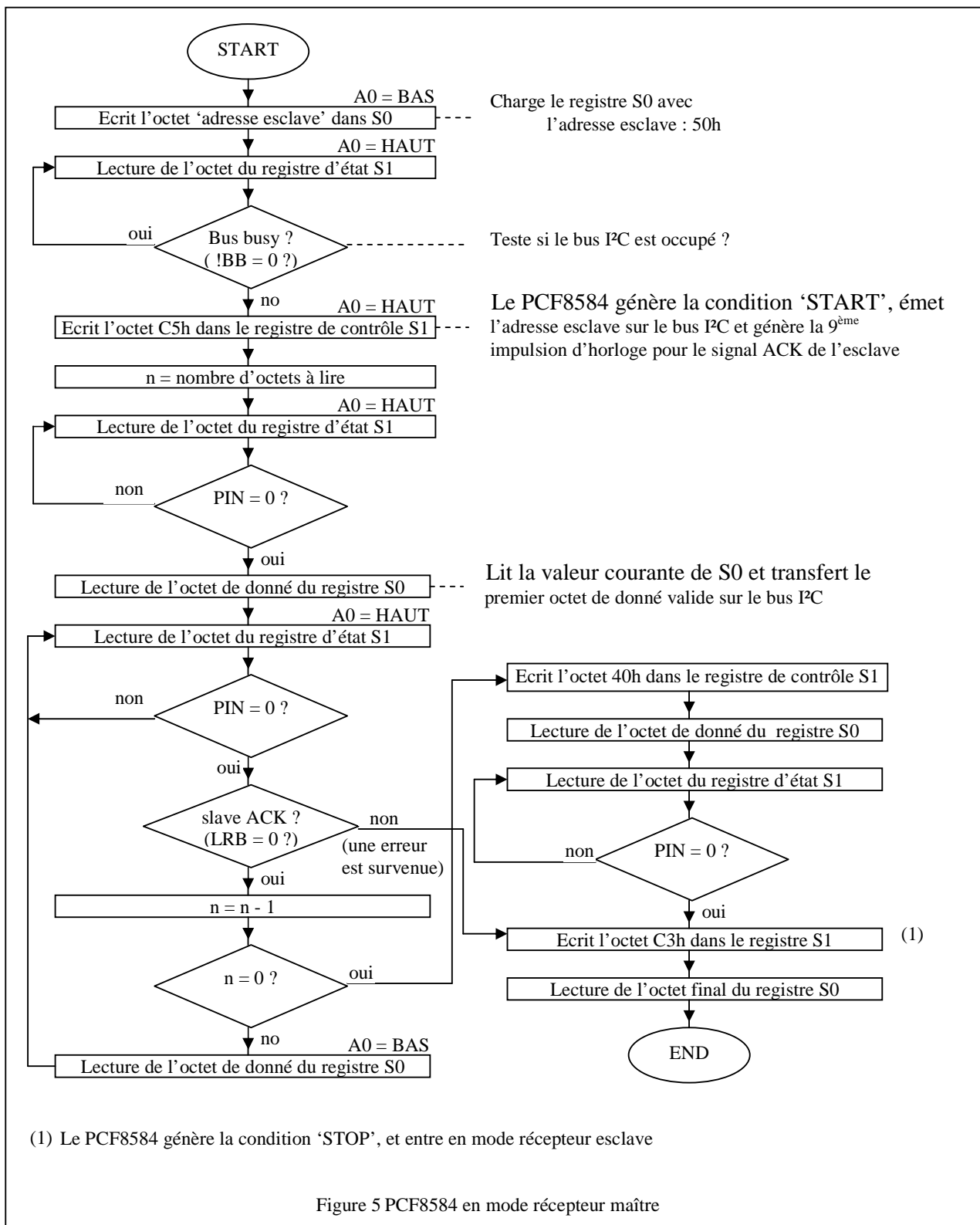
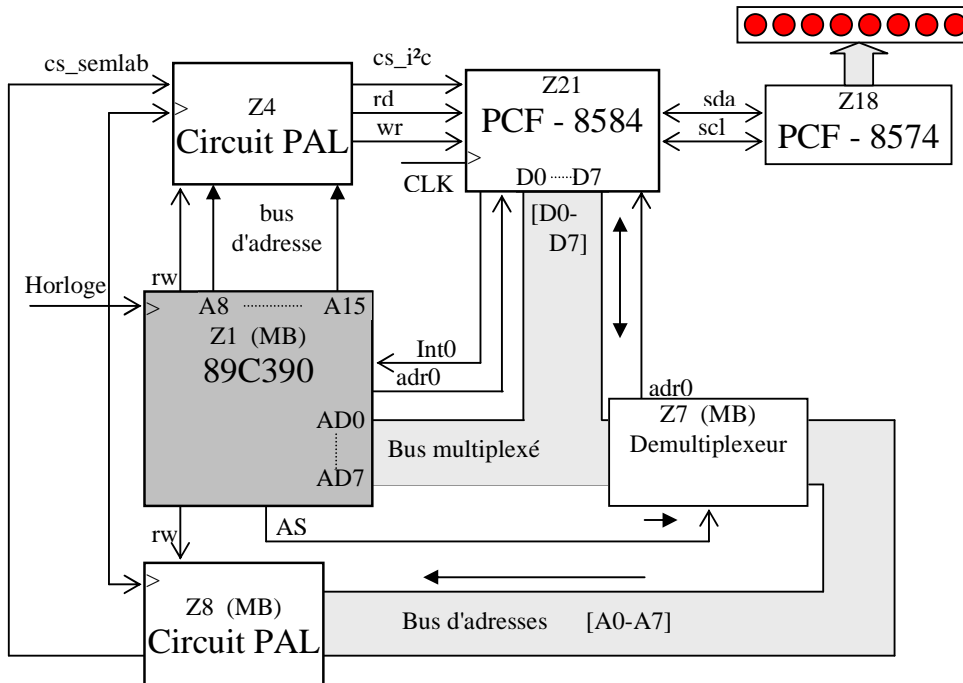


Figure 4 PCF8584 en mode émetteur maître



## IX a Port d'entrées-sorties 8 bits programmable ( PCF8574 )

### 1. Liaison microcontrôleur vers le port d'entrées/sorties 8 bits programmable.



(MB) : Composants se trouvant sur la carte mère MC51TT  
Les autres composants se trouvent sur la carte SemLab1-B

### 2. Descriptif du fonctionnement

Le circuit PCF8574 permet d'utiliser un port de 8 bits configurables individuellement en entrée ou en sortie à partir du bus I<sup>2</sup>C.

Ses sorties lachées avec une haute conduction en courant permet de relier directement des leds.

Lorsque des données doivent être échangées entre le microcontrôleur et ce circuit, celles-ci passent par une liaison de type I<sup>2</sup>C.

Pour communiquer avec le circuit, il faut envoyer, via la liaison I<sup>2</sup>C, l'adresse de ce composant pour le sélectionner. Il dispose de 3 broches d'adresses qui permettent une utilisation pouvant aller jusqu'à 8 composants. Il existe deux adresses différentes en fonction du type de boîtier (A ou AP).

Adresse du composant PCF8574 :

0	1	0	0	A2	A1	A0	R/W	PCF8574P
0	1	1	1	A2	A1	A0	R/W	PCF8574AP
Bits fixés par le constructeur				0	1	0	Signal lecture/écriture	
				Câblage sur la carte SemLab1-B				

Il y a donc 4 adresses pour communiquer avec le PCF8574

- Adresse 44h, écriture dans un des registres du PCF8574P
- Adresse 45h, lecture dans un des registres du PCF8574P
  
- Adresse 74h, écriture dans un des registres du PCF8574AP
- Adresse 75h, lecture dans un des registres du PCF8574AP

### 3. Descriptif du programme P\_I2C\_E-S.A51

Rôle : afficher l'état des sorties du PCF8574 sur les 8 leds prévues à cet effet.

Ce programme permet de communiquer avec le circuit PCF8574 par l'envoi de commandes. Les échanges de données entre le microcontrôleur et le port d'entrées/sorties programmable s'effectuant au moyen du bus I<sup>2</sup>C. L'avantage de ce programme est de tester si le bus I<sup>2</sup>C a bien été configuré car une mauvaise configuration du bus I<sup>2</sup>C se ressentira au niveau du défilement du chenillard.

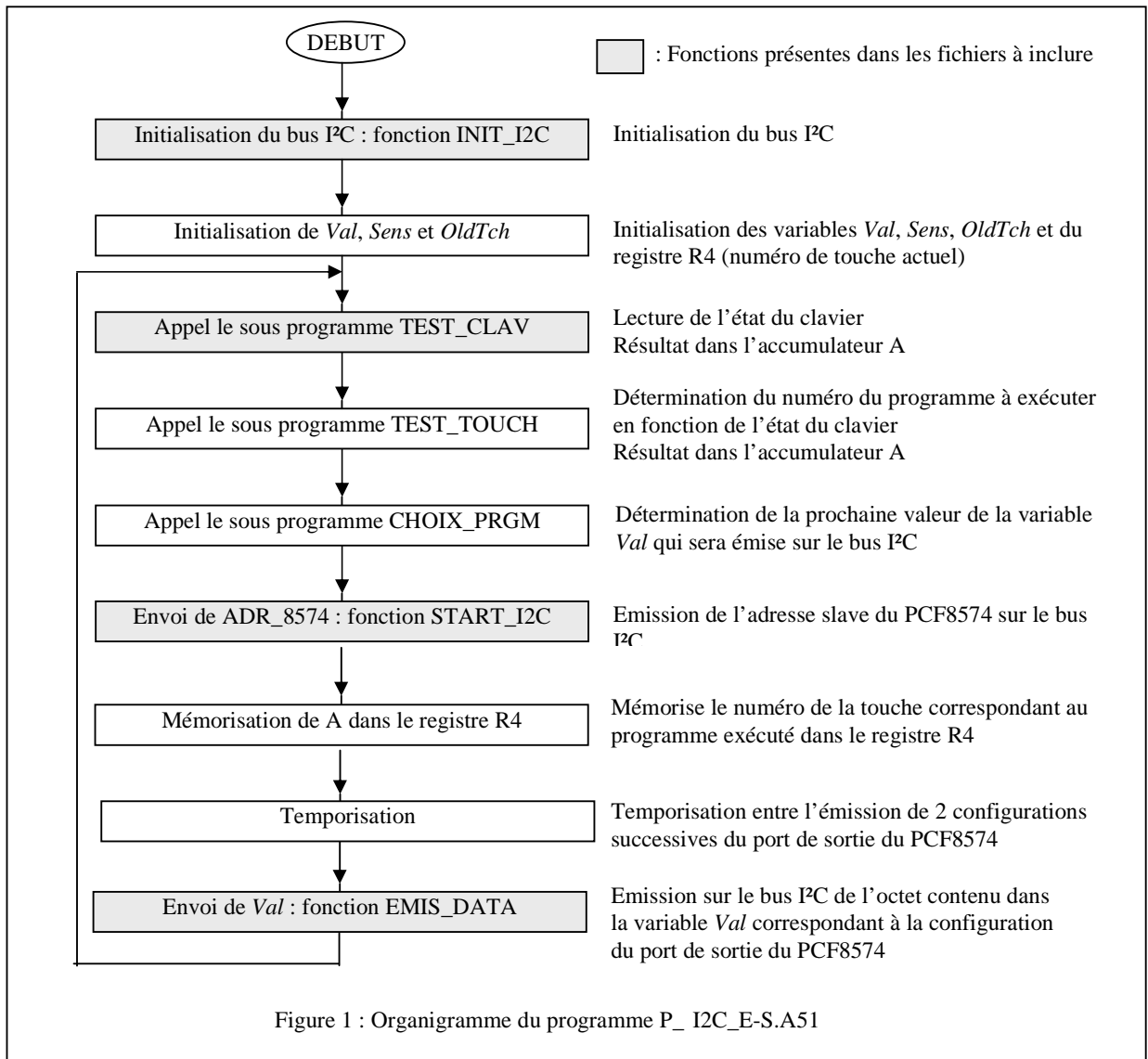
Les fonctions permettant d'utiliser le bus I<sup>2</sup>C se trouvent dans le fichier "S\_I2CP84.A51". Celui-ci est à inclure dans le programme.

Ce programme dans une boucle infinie, lit l'état du clavier, détermine le programme à exécuter parmi les six et émet la valeur correspondant sur le bus I<sup>2</sup>C.

#### 3.1 Organigramme (Figure 1 )

#### 3.2 Sous programme TEST\_CLAV

Le sous programme TEST\_CLAV qui se trouve dans le fichier à inclure « S\_CLAVIER.A51 » permet de lire l'état du clavier téléphonique et de retourner le code correspondant par l'intermédiaire de l'accumulateur A. Vous trouverez plus d'explication dans le chapitre concernant le clavier téléphonique.



### 3.3 Sous programme TEST\_TOUCH

Le sous programme TEST\_TOUCH permet d'affecter un nombre allant de 1 à 6 pour les codes du clavier renvoyés par le sous programme TEST\_CLAV, et 0 pour les autres cas. Le résultat qui se trouve dans l'accumulateur A sera par la suite mémorisé dans le registre R4. La correspondance est indiquée dans le Tableau 2.

Tableau 2 : Sous programme TEST\_TOUCH

CODE DU CLAVIER	TOUCHE CORRESPONDANTE	R4
00	AUCUNE	0
44	1	1
42	2	2
41	3	3
24	4	4
22	5	5
21	6	6
AUTRE	X	0

### 3.4 Sous programme CHOIX\_PRGM

Le sous programme CHOIX\_PRGM à pour fonction de déterminer la valeur de la variable *Val* qui contient l'octet à envoyer au PCF8574 pour l'affichage des leds.

Le déroulement du sous programme est le suivant :

Sélection du programme à exécuter en fonction du registre R4 (Tableau 3). Si le contenu du registre R4 est 00h, il est remplacé par celui de la variable *OldTch* qui contient le numéro du programme précédemment exécuté.

Le code d'un des six programmes est exécuté ; un test réalisé sur la variable *OldTch* permet de savoir s'il s'agit du programme qui a été exécuté précédemment. Si tel n'est pas le cas, la variable *Val* et éventuellement la variable *Sens* et le registre R3 (pour le programme 6) sont réinitialisées.

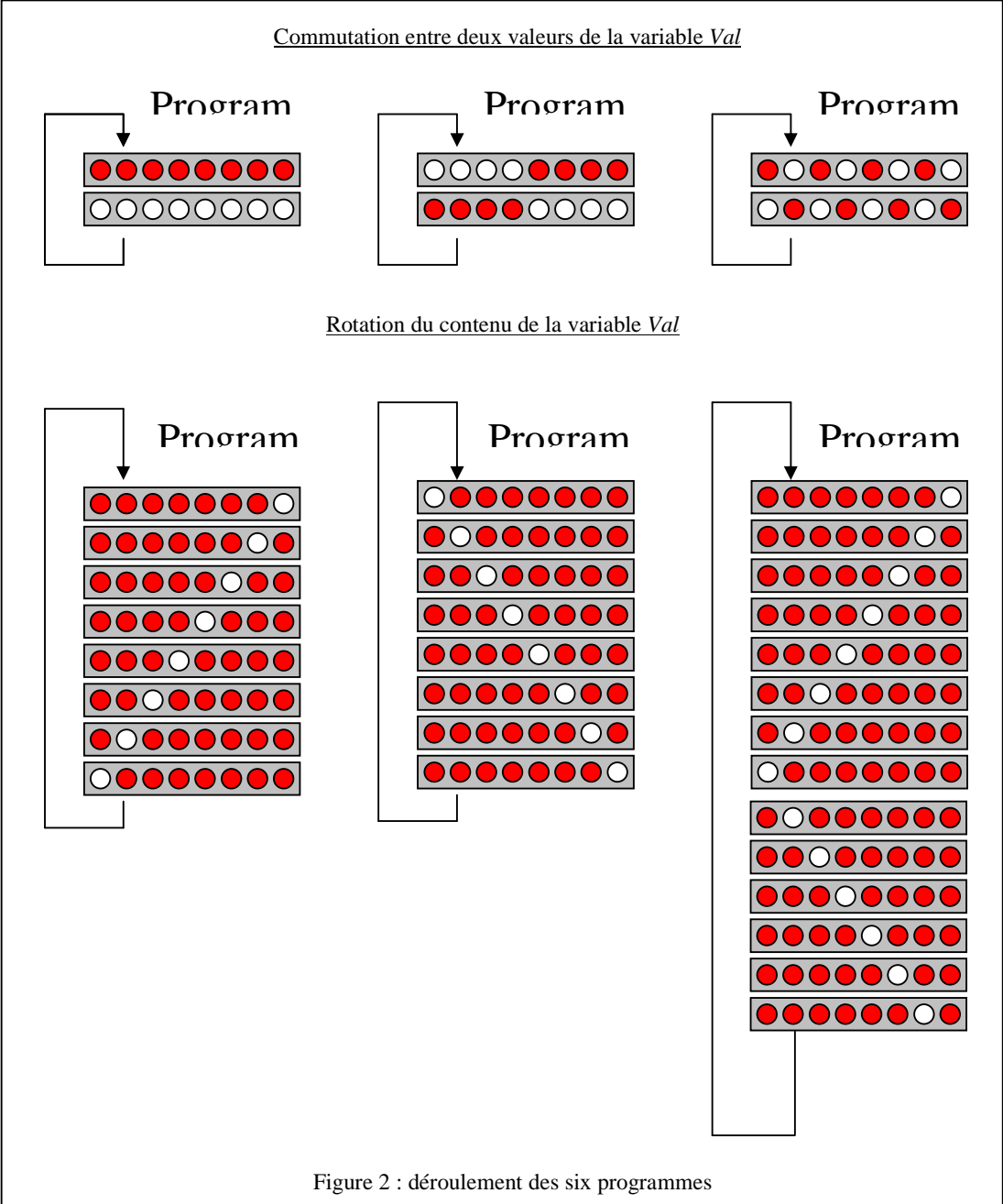
Après l'exécution du programme, la variable *Val* contient la nouvelle valeur à envoyer au PCF8574.

Le numéro du programme exécuté est placé dans l'accumulateur A avant d'être mémorisé par la suite dans la variable *OldTch*.

Le déroulement des six programmes est indiqué figure2.

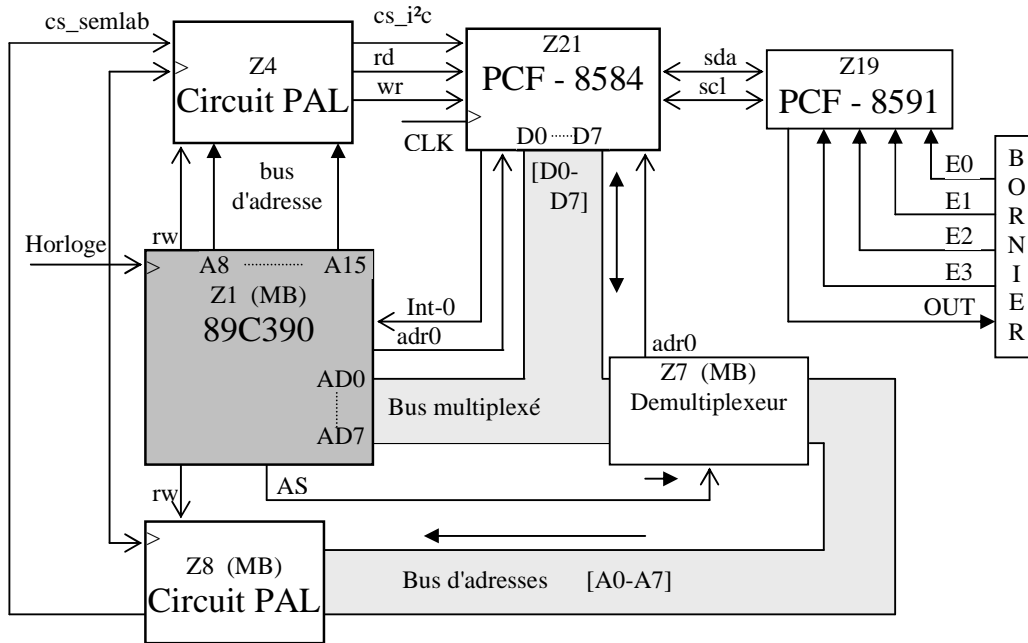
Tableau 3 : Sous programme CHOIX\_PRGM

NumTch	PROGRAMME EXECUTE
00	Programme précédent
01	Programme 1
02	Programme 2
03	Programme 3
04	Programme 4
05	Programme 5
06	Programme 6



# IX b - Convertisseur AD / DA ( PCF8591 )

## 1. Liaison microcontrôleur vers le convertisseur AD / DA .



(MB) : Composants se trouvant sur la carte mère MC 51TT  
 Les autres composants se trouvent sur la carte SemLab1-B

## 2. Descriptif du fonctionnement

Le circuit PCF8591 (convertisseur AD / DA)

Ce circuit d'acquisition de données dispose de quatre entrées analogique, d'une sortie analogique, ainsi que d'une interface avec le bus série I<sup>2</sup>C.

Pour communiquer avec le circuit, il faut envoyer, via la liaison I<sup>2</sup>C, l'adresse de ce composant pour le sélectionner. Trois broches d'adresse sont utilisées pour programmer les adresses matérielles, permettant d'utiliser jusqu'à huit composants connectés au bus I<sup>2</sup>C sans câblage supplémentaire.

Adresse du composant PCF8591 :

1	0	0	1	A2	A1	A0	R/W
				0	0	1	
Bits fixés par le constructeur				Câblage sur la carte SemLab1-B		Signal lecture/écriture	

Il y a donc 2 adresses pour communiquer avec le PCF8591

- Adresse 92h , écriture dans un des registres du PCF8591
- Adresse 93h , lecture dans un des registres du PCF8591

### 3. Descriptif du programme P\_ANALOG.A51

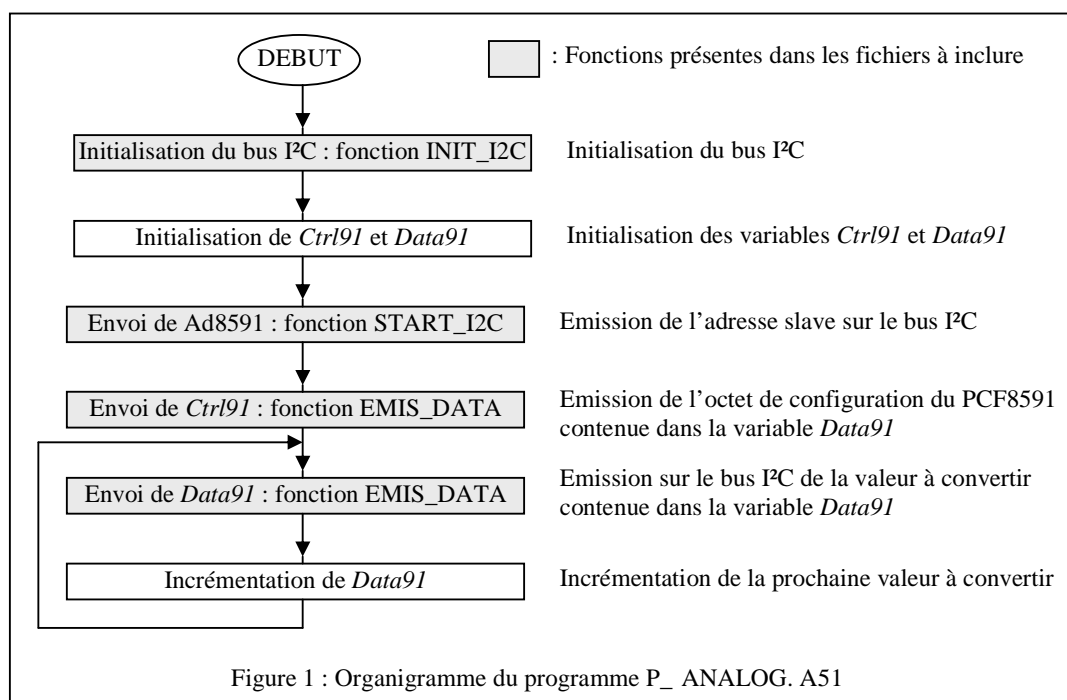
Rôle : Générer une rampe sur la sortie analogique du PCF8591

Ce programme permet de communiquer avec le circuit PCF8591 par l'envoi de commandes. Les échanges de données entre le microcontrôleur et le circuit d'acquisition de données s'effectuant au moyen du bus I<sup>2</sup>C.

Les fonctions permettant d'utiliser le bus I<sup>2</sup>C se trouvent dans le fichier "S\_I2CP84.A51". Celui-ci est à inclure dans le programme.

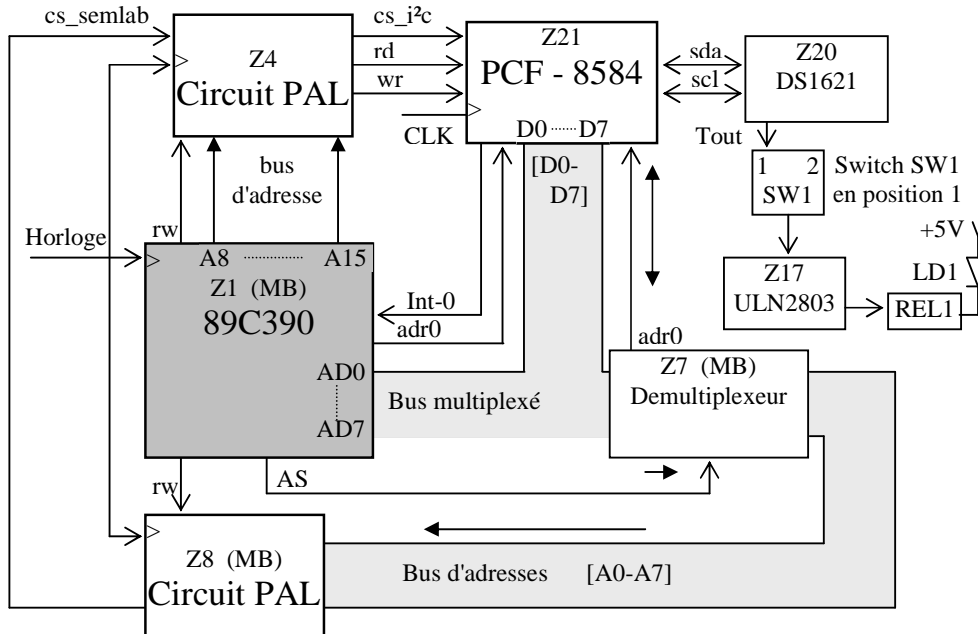
Ce programme initialise le contrôleur de bus I<sup>2</sup>C, les variables du programme et le circuit d'acquisition de données puis dans une boucle infinie, émet la valeur correspondant à la rampe sur le bus I<sup>2</sup>C.

#### 3.1 Organigramme du programme principal ( Figure1 )



## IX c - Capteur de température ( DS1621 )

### 1. Liaison microcontrôleur vers le capteur de température .



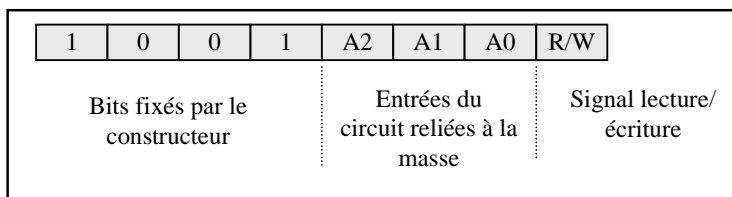
(MB) : Composants se trouvant sur la carte mère MC51TT  
 Les autres composants se trouvent sur la carte SemLab1-B

### 2. Descriptif du fonctionnement

Le circuit DS1621 (capteur de température)

Ce circuit permet de savoir quelle est la température ambiante, ainsi que de piloter une sortie thermostat de type tout ou rien. Lorsque des données doivent être échangées entre le microcontrôleur et ce circuit, celles-ci passent par une liaison de type I<sup>2</sup>C. Pour communiquer avec le capteur, il faut envoyer, via la liaison I<sup>2</sup>C, l'adresse de ce composant pour le sélectionner.

Adresse du composant DS1621 :



Il y a donc 2 adresse pour communiquer avec le DS1621  
 - Adresse 90h , écriture dans un des registres du DS1621  
 - Adresse 91h , lecture dans un des registres du DS1621

### 3. Descriptif du programme P\_CaptTemp.A51

Rôle : afficher la température du circuit DS1621 sur un afficheur LCD.

Ce programme permet de configurer le circuit DS1621 par l'envoi de commande. Les échange de données entre le microcontrôleur et le capteur de température s'effectuant au moyen du bus I<sup>2</sup>C.

Le tableau TabOctet contient les commandes à envoyer au capteur de température.

TabOctet     db     0EEh,AAh,0ACh,09h,41h,42h

0EEh Start Convert            lance une conversion de la température  
0AAh Read Temperature        lecture de la température  
0ACh Access Config            accès au registre de configuration  
09h    Octet à écrire dans le registre de configuration

Celui-ci peut être remplacé par l'octet 0Bh pour changer le niveau logique de l'état actif de la sortie du thermostat

41h    Access TH                accès au registre TH du thermostat  
42h    Access TL                accès au registre TL du thermostat

Les fonctions permettant d'utiliser le bus I<sup>2</sup>C se trouvent dans le fichier "S\_I2CP84.A51", celles qui permettent de gérer l'afficheur LCD se trouvent dans le fichier "S\_AFILCD.A51". Ces deux fichiers sont à inclure dans le programme.

Ce programme configure le capteur de température, puis dans une boucle infinie, lance une conversion, lit la valeur de la température, met en forme cette valeur avant de l'afficher sur l'afficheur LCD.

#### 3.1 Organigramme du programme principal     (Figure 1 )

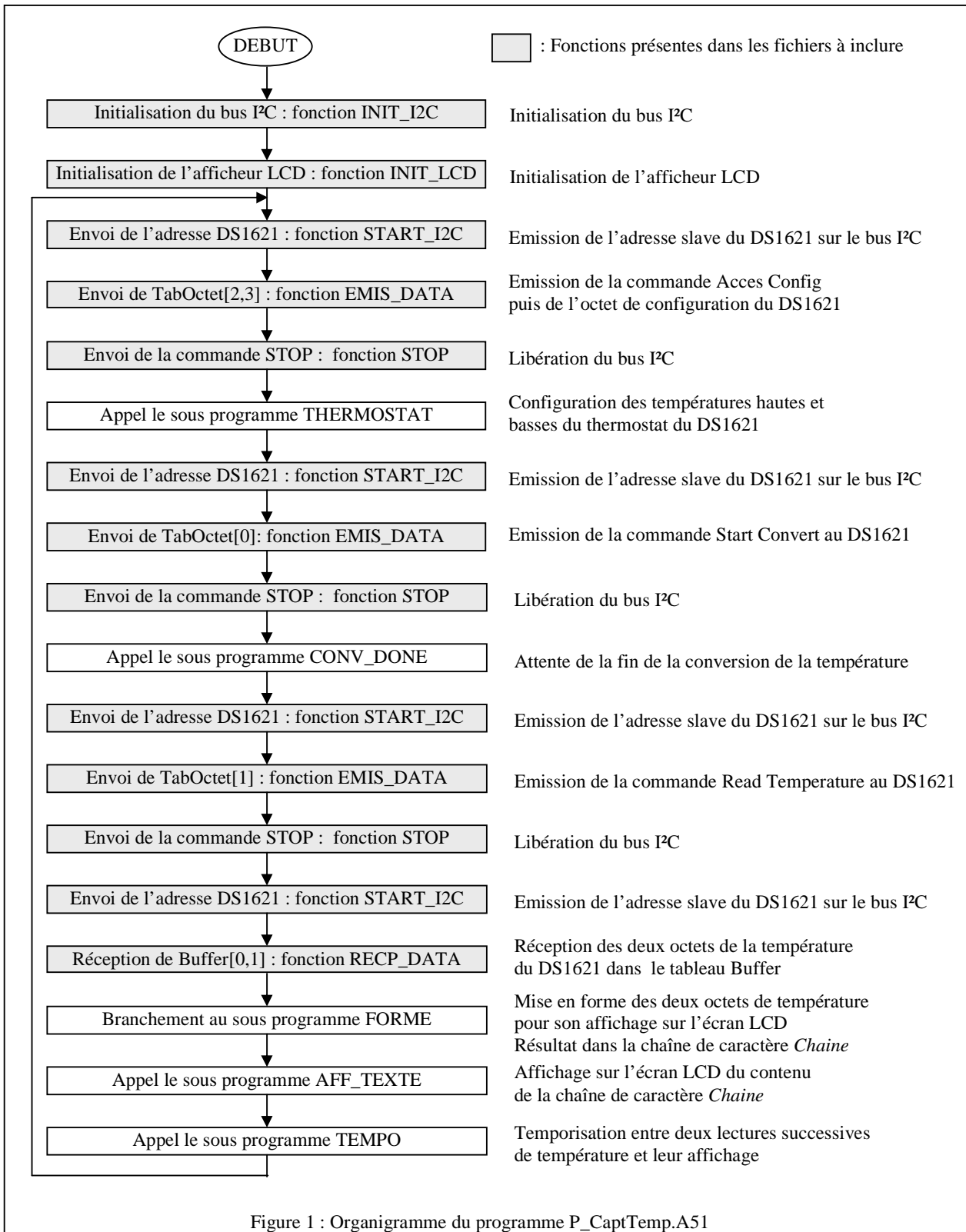
#### 3.2 Configuration du DS1621

Pour configurer le circuit DS1621, il faut écrire dans son registre de configuration l'octet 09h (ou bien 0Bh : voir le paragraphe 3).

L'accès au registre de configuration se fait par l'envoi de la commande *Access Config*.

DONE	THF	TLF	NVB	1	0	POL	1SHOT
0	0	0	0	1	0	0	1

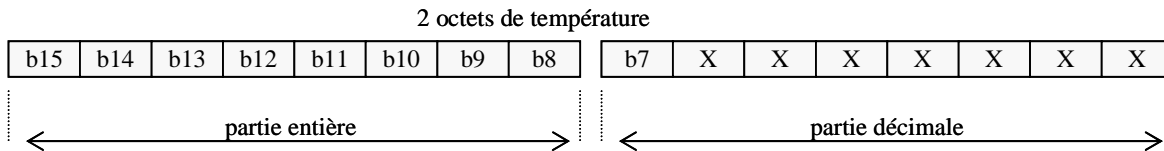
Exemple d'écriture dans le registre de configuration



### 3.3 Configuration du thermostat (sous programme THERMOSTAT)

Le thermostat du DS1621 est à hystérésis, c'est à dire que la température d'activation du thermostat peut être différente de celle qui le désactive. Il faut donc configurer deux registres, TH qui contient la température qui active le thermostat et TL, qui contient la température qui le désactive.

Pour configurer les registres TH ou TL, il faut d'abord envoyer la commande d'accès à l'un de ces deux registres puis écrire deux octets à la suite contenant la valeur de la température en respectant la forme suivante :



Pour la correspondance entre cette forme et la valeur décimale de la température, se référer au tableau 1.

Il est possible de choisir le niveau logique de l'état actif ou inactif de la sortie du thermostat à partir du bit 'POL' du registre de configuration. Un '1' correspond à la sortie active niveau haut, et un '0' correspond à la sortie active niveau bas.

Pour connaître l'état de la sortie du thermostat, il faut placer l'interrupteur SW1 en position 1 et visualiser la led LD1.

### 3.4 Attente de fin de conversion (sous programme CONV\_DONE)

Une conversion est effectuée à chaque fois qu'un ordre *Start Convert* est reçu. Le bit DONE passe à '1' à chaque fin de conversion.

DONE	THF	TLF	NVB	1	0	POL	1SHOT
X	0	0	0	1	0	0	1

↑  
 '0' conversion en cours du DS1621  
 '1' conversion finie

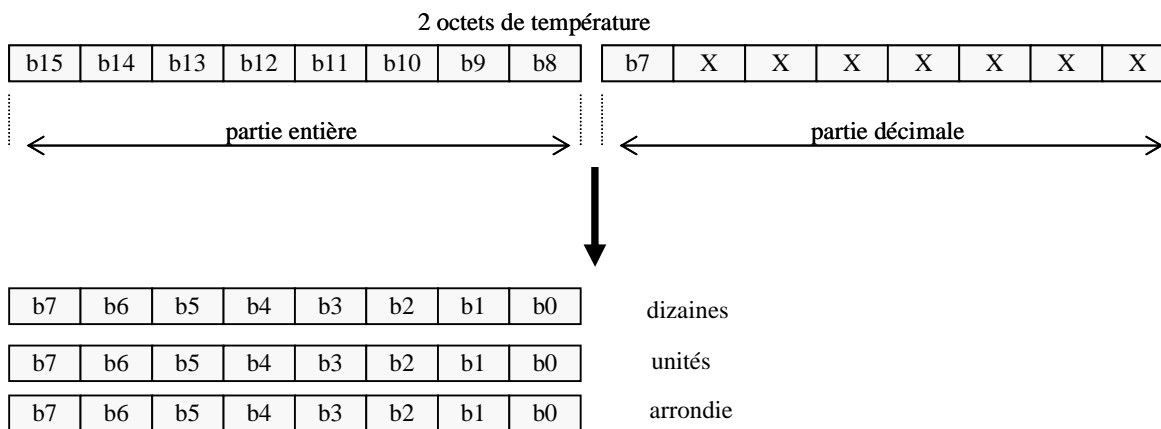
### 3.5 Lecture de la température

Après avoir envoyé la commande *Read Temperature*, les deux prochains octets qui seront lus du DS1621 correspondront à la partie entière puis à la valeur de l'arrondi de la température.

TEMPERATURE	DIGITAL OUTPUT (Binaire)	DIGITAL OUTPUT (hexadécimal)
+125°C	01111101 00000000	7D00h
+25°C	00011001 00000000	1900h
+0.5°C	00000000 10000000	0080h
+0°C	00000000 00000000	0000h
-0.5°C	11111111 10000000	FF80h
-25°C	11100111 00000000	E700h
-55°C	11001001 00000000	C900h

### 3.6 Mise en forme de la température (sous programme FORME )

La température se trouvant sous forme de deux octets, il s'agit de traiter ces deux octets pour en dégager les chiffres des dizaines, des unités ainsi que de l'arrondi.



### 3.7 Affichage de la température

Utilisation de la fonction AFF\_TEXTE qui permet d'afficher le contenu d'une chaîne de caractère ASCII se terminant par la valeur 0, sur l'afficheur LCD.

Pour convertir une valeur numérique hexadécimale en ASCII, il suffit de lui ajouter la valeur hexadécimale 30h.

Exemple :

hexadécimal	code ASCII (valeur hexadécimale + 30h)
0h	30h
4h	34h
9h	39h