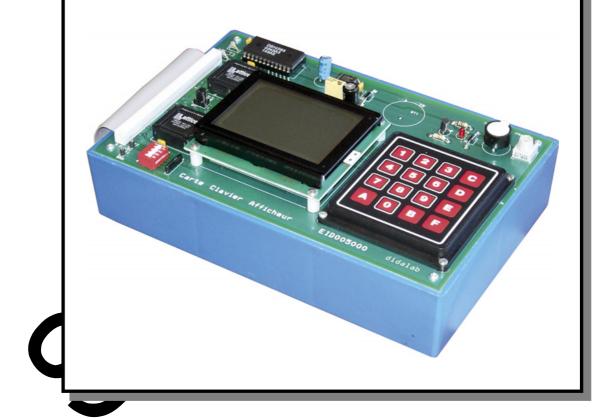
Carte Clavier Afficheur graphique Horloge temps réels

EID005



Auteur: KOMA N'Gally Professeur BTS IRIS IFA Delorozoy CCI Versailles

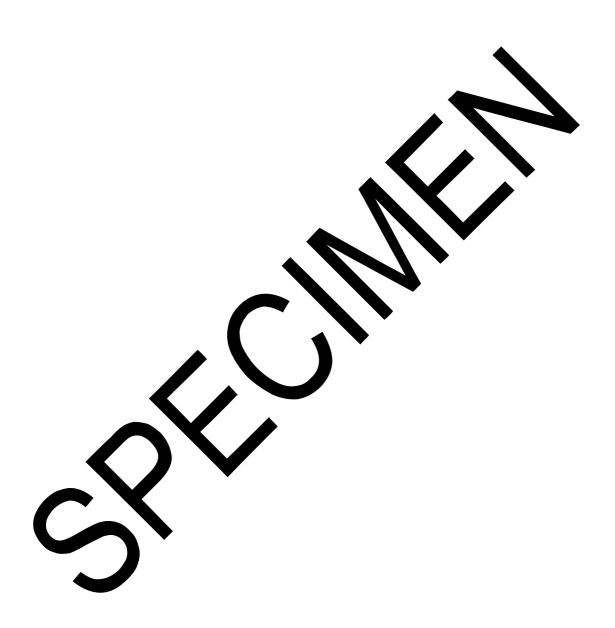


DIDALAB

5 Rue du Groupe Manoukian 78990 Elancourt Tel: 01.30.66.08.88 / Fax: 01.30.66.72.20 ge@didalab.fr

23/08/2007 EID215040



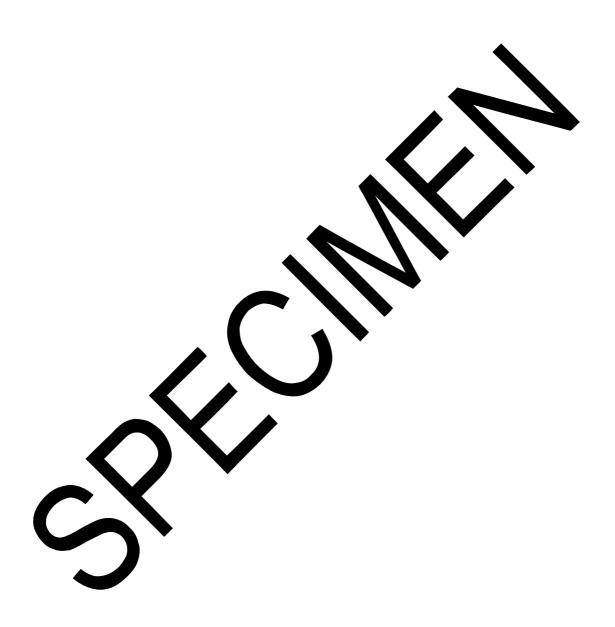




SOMMAIRE

SUR L'AFFICHEUR	IEXIE 5
1.1 Enoncé du sujet	5
1.2 Eléments de solution	6
1.2.1 Description sommaire de l'afficheur	6
1.2.2 Programme en C	15
1.2.3 Programme en Assembleur	17
1.2.3 Programme en Assembleur EID005_TP 2 ECRITURE D'UNE CHAINE DE CARA TERES E MODE	TEXTE
SUR L'AFFICHEUR	22
2.1 Enoncé du sujet	22
2.2 Eléments de solution	23
2.2.1 Description sommaire de l'afficheur	23
2.2.2 Programme en C	33
2.2.2 Programme en C	34
EID005_TP 3 LECTURE D'UN CLA 'ER N. 17N. 'SE EN MODE POLLING	
3.1 Enoncé du sujet	39
3.2 Eléments de solution	40
3.2.1 Description somm are du clavier	40
3.2.2 Programme en C. 3.2.3 Programme Assembleur	44
3.2.3 Programme Assembleur	46
EID005_TP 4 DETP TION D'A TIVAT ON D'UNE TOUCHE ET LECTUR	KE EN
MODE POLY IG	
4.1 Enoncé du stat	53
4.2 Eléments a solution 4.2.1 escrip on solution et du clavier	54 54
4.2.1 Pescriptori solutione du davier	54
4.2.2 Programme en C	50
EIF 05 YP 5 ESSIN DE LIGNES, CERCLES ET COURBES SUR LE LA	60
5.1 Ep 2 du s'et	6969
5.2 Temen's de solution	70
5.2.1 De ription de l'afficheur en mode graphique	
5.2.2 gramme principal	
5.2.3 Organigramme du tracé d' une sinusoïde sur un oscilloscope	73
5.2.4 Programme en " C " :	75
EID005 TP 6 DESSIN D'UNE HORLOGE SUR L'ECRAN GRAPHIQUE	
6.1 Eléments de solution	79
6.1.1 Définition géométrique de l'horloge	79
6.1.2 Programme principal	
6.1.3 Organigramme	
6.1.4 Programme en C	





Ref: EID215040

EID005_TP 1 ECRITURE D'UNE CHAINE DE CARACTERES EN MODE TEXTE SUR L'AFFICHEUR

1.1 Enoncé du sujet

Objectifs :	Etre capable d'utiliser les utilitaires rangée en biblio èque, permettant la gestion de l'afficheur LCD y 30mq 14 164 pixels.
Cahier des charges :	Sujet Ecrire un programme et asset l'agur et en C qui réalisera l'affichage d'une cha îne de caractères. La longur a maximale le la chaîne est de 128 caractères. On dont era les coordonniées du 1 ^{er} caractère.

l'atériel nécessaire :

Microprdinateur de type PC sous Windows 95 ou ultérieur, Cart mère 10192 blu à microcontrôleur 68332, Réf : EID210000 Carte Stater Afri heur Horloge Temps réel : EID005000 Câble de liaison SB, ou à défaut câble RS232, Réf : EGD000003 Alimentation AC 8V, 1 A Réf : EGD000001,

Documentations nécessaires :

Document: DMS Carte Clavier Afficheur Horloge Temps réel: EID00500 Application Notes for the T6963C LCD Graphics Controller Chip (TOSHIBA) T6963c DOT MATRIX LCD CONTROL LSI (TOSHIBA)

Durée : 1 séance de 3 heures



1.2 Eléments de solution

1.2.1 Description sommaire de l'afficheur

1.2.1.1 Représentation de l'afficheur LCD 128x64

Attention:

Pour des raisons de conformité avec la documentation du constructeur, les variables x et y représentent respectivement l'ordonnée (verticale) et l'abscisse (horizontale). Le point x = 0, y = 0 est en haut à gauche et le point x = 63, y = 127 en bas à droite de l'écran du LCD.

Le contrôleur T6963C dispose d'une mémoire de 8 ko.

Mode texte fig.1

Dans l'étude qui suit, la zone texte est placée de l'adres e 0000 à 007 de la mémoire écran (VRAM), soit 128 caractères.

Octet de poids faible : 00 toujours fixe

Octet de poids fort : 00 à 7F en hexadécimal.

Le quartet de poids fort de cet octet désigne le nume de la colonge y. Le quartet de poids faible désigne le nume de la colonge y.

Exemple

Le caractère numéro 59 est pla é en : x = 3, y = 1; ce qui donne en hexadécimal les valeurs : x = 3, y = 1 d'où el mémoire, le octets suivants pour le paramètre TH (Text Home Adress :

Adress ower (59 en décimal)

Adress upper = 0x00

MODE TEXTE

				•												•
C		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127

fig.1





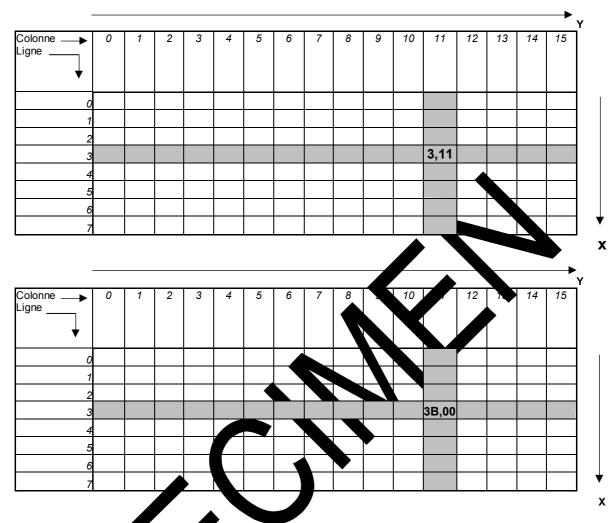


fig.1 suite

Générateur de Laract res la erne CG ROM)

Le générateur se actères interne utilise un code ASCII décalé de 0x20 ; exemple : la lette Acodée 41 en ASCII est représentée par la valeur 0x21 dans le T6963C (voir ableau ci-dess us).

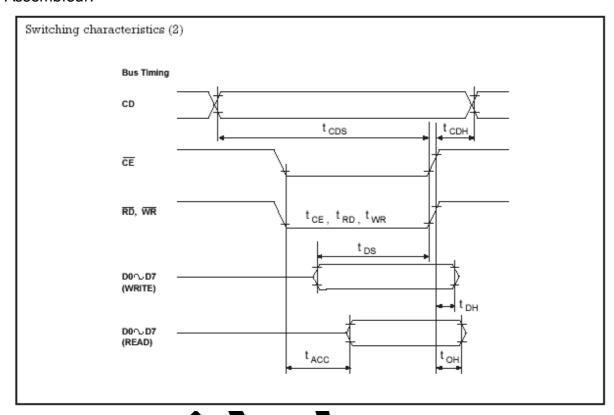
Cela ignit que pour énvoyer un caractère ASCII, il faut soustraire de son code, la valeur 0x20.

3



Chronogrammes Ecriture / Lecture de données ou commandes

Ces chronogrammes doivent être générés pour chaque accès au LCD. Ils sont réalisés et décrits en détail notamment dans les sous-programmes en Assembleur.



La génération de cha un de control de character de correspondant.

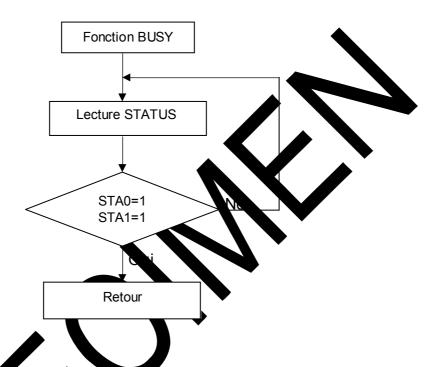




1.2.1.2 Gestion de l'afficheur LCD

Avant chaque mouvement (écriture ou lecture) donnée (commande ou donnée) entre l'afficheur et le processeur de commande, il faut s'assurer que le LCD est prêt à exécuter l'opération.

D'où la nécessité de commencer par tester les bits d'état (ou de statut) de son registre d'état : STA0 et STA1 ; ce qui donne l'organigramme suivant :



1.2.1.3 Instruct as de l'as heur

Les instruction de l'fich y sont connées dans le tableau ci-dessous.

Les organigns ames d'dessous définissent les modes d'écriture d'instruction de commande nécessaint 2, 1 ou 0 octets.

Les rganier des vivants définissent en détail les modes d'écriture et de lecture des containdes et des données.



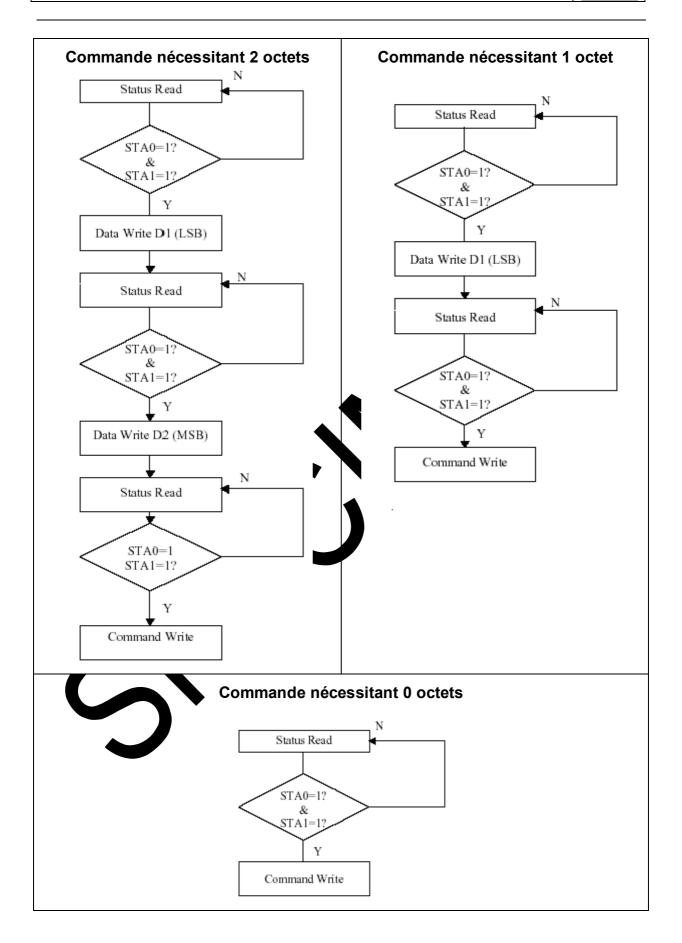
T6963C Instruction Set

Commands	D7	D6	D5	D4	D3	D2	D1	D0	Description	Execute Time
Pointer Set	0	0	1	0	0	N2	N1	N0		Status
						0	0	1	Cursor Pointer Set	check
						0	1	0	Offset Register Set	1
						1	0	0	Address Pointer Set	
Control Word	0	1	0	0	0	0	N1	N0		32 x 1/fosc
Set Commands	\vdash						0	0	Text Home Address Set	Inose
Det Communes	-	\vdash					0	1	Text Area Set	1
							1	0	Graphic Home Address Set	1
			\vdash				i	1	Graphic Area Set	1
Mode Set	1	0	0	0	CG	N2	N1	NO	Chapme Area Sec	32 x
Mode Det		u u		, a		112	141	110		1/fosc
	-	_	_		0		<u> </u>	_	CG ROM Mode	-
	\vdash		_		l	_	_		CG RAM Mode	4
						0	0	0	"OR" Mode	4
	\vdash	_				0	0	1	"EXOR" Mode	-
	$\vdash \vdash$		<u> </u>			0	1	1	"AND" Mode	4
			_			1	0	0	Text only (attribute capability)	-
Display Modes	1	0	0	1	N3	N2	N1	N0		32 x 1/fosc
					0				Graphics Off	1
					1		\vdash		Graphics On	1
						0			Text Off	1
						1			Text On	1
							0		Cursor Off	1
	\vdash						1		Cursor On	1
	-						$\overline{}$	0	Cursor blink Off	1
								1	Cursor blink On	1
Cursor Pattern	1	0	1	0	0	N2	N1	N0	N2-N0: No. of lines for cursor +1	32 x
		ů	·	, a						1/fosc
Select	-	_	_			0	0	0	Bottom Line cursor	4
	\vdash		_			0	0	1	2 line cursor	4
	\vdash		_				<u> </u>	<u> </u>		4
	_	_	_	_		1	1	1	8 line cursor (block cursor)	
Data Auto	1	1	0	0	0	0	N1	No		32 x 1/fosc
Read/Write							0	0	Data Auto Write Set	1
	-						0	1	Data Auto Read Set	1
	-						1	0	Auto reset (Address pointer auto-	1
									incremented) for continuous rd/wr	I
Data Read/Write	1	1	0	0	0	N2	N1	N0		
						0			Address Pointer up/down	1
						1			Address Pointer unchanged	1
							0		Address Pointer up	1
							1		Address Pointer down	1
								0	Data Write	1
								1	Data Read	1
Screen Peeking	1	1	1	0	0	0	0	0	Read Displayed Data	Status
Screen Copy	1	1	1	0	1	0	0	0	Copies 1 line of displayed data whose	Status
(Note 3)			1	_	^		l		address is indicated by the Address	check
v,							l		Pointer to Graphic RAM area	
Bit Set/Reset	1	1	1	1	N3	N2	N1	N0	N2-N0 indicates the bit in the pointed	Status
- A Country of the Co	•	•	1	•	1.0			``"	address	check
					0				Bit Reset	1
					1				Bit Set	1
					<u> </u>	0	0	0	Bit 0 (LSB)	1
	$\vdash \vdash$		\vdash		_	0	0	1	Bit 1	1
		-	\vdash			1	Ī	Ť		1

Note:

1. * = DONT CARE

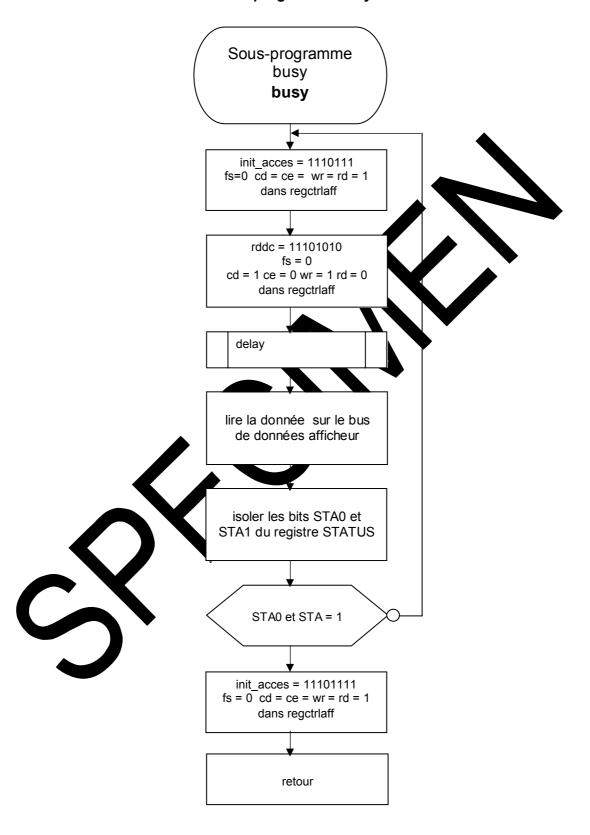






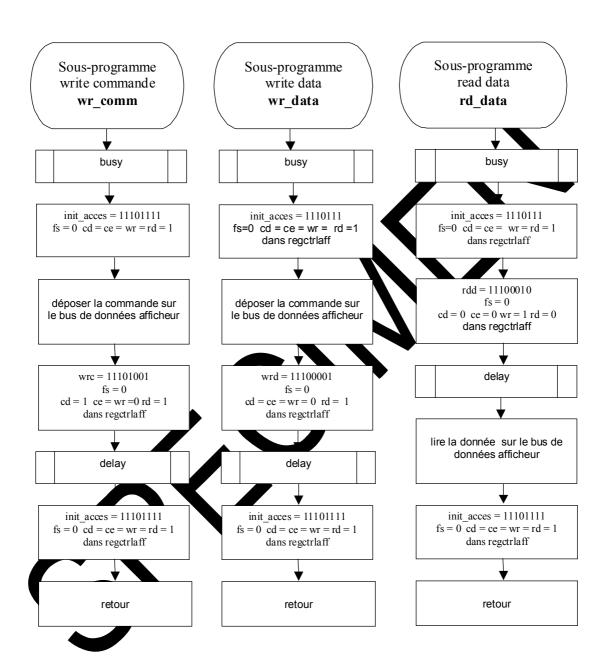
Organigrammes détaillés des sous-programmes principaux

Sous-programme busy





Sous-programmes écriture lecture





1.2.1.4 Organigramme principal

Vous disposez des utilitaires suivants avec leurs commentaires.

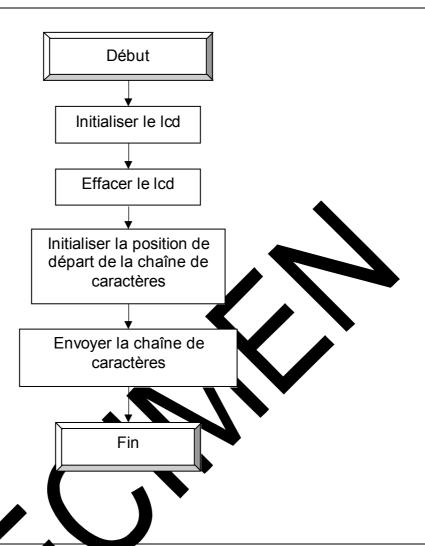
Utilitaires	Commentaires
<pre>void init_aff()</pre>	Initialisation des
	paramètres TH, TA, GH
	GA Mode
<pre>void lcd_cls()</pre>	Effacement de l'écran
<pre>void lcd_write_command(unsigned char commande)</pre>	Ecriture d'une
	commande
<pre>void lcd_write_data(unsigned char data)</pre>	Ecriture une donnée
	Définition de la
<pre>void lcd_gotoxy(unsigned char px, unsigned char py)</pre>	po d'un
	cara ere
	pixel
	px et py cont les
	dor Zes la er et
	y per.
<pre>void lcd_out_str(char *texte)</pre>	znvoi d'a le chaîne de
	ract les pontée par
	la / lable * texte

Important:

Pour utiliser ses fonctions ainsi que le clavier afficheur avec le compilateur C/C++, il faut configurer l'ulinker du logiciel eid210. Pour cela : aller dans le mena configuration puis cliquer gnu C/C++. Ensuite aller dans l'ongle *linker*, clipur sur *ajouter*, sélectionner le fichier « EID005_La co » etalinir en cliquant sur ouvrir.

Ref: EID215040





1.2.2 Programme en **G**



```
//==========
// FONCTION PRINCIPALE
//============
main()
init_aff();
                    // Initialisation de l'afficheur
                        Effacement de l'afficheur
lcd cls();
                    //
lcd gotoxy(0x10,00);
                        Définition de la position de départ de la
                    // Chaîne
// Envoi de la chaîne de caractères
lcd_out_str ( "EID005000 : CLAVIER, AFFICHEUR,
                                                        ORLOGE
TEMPS REEL");
// Les espaces dans le texte permettent d'éviter de
//
     Fin du programme
```

Ref: EID215040

1.2.3Programme en Assembleur

```
TP SUR LA CARTE EID005
     Ecrire un programme en Assembleur
     et en C et qui réalise la lecture d'un touche
     sur un clavier matricé 4 x 4, en mode pooling
    ***********
     Nom du FICHIER : EID005 TP1.src
     *****
**********
     La commande ou la donnée à écrire sont d'abor
                                                         ans Do
     La donnée lue est rangée dans D0
* *
     La position x, y d'un caractère est placée
                                                     ement
                                                            ans DO, D1
                                                   lacée dans A0
     L'adresse de début d'une chaîne de cara
*********
     Bits du registre de contrôle de l
     ctrlaff :
               b0=rd
                b1=wr
                b2=ce
                b3=cd
                b4=fs
 Inclusion du fichier défir
                           sant les
                                     ifférents labels
* de l'EID210
     include EID21
                   def
     section va
     Défin
                            hysiques de l'afficheur et du clavier
            on de
                   adr
eid005
                 30000
                           * Adresse de base Carte clavier afficheur rtc
                eid005+3
ctrla
                           * Registre contrôle afficheur : bus de
                           * contrôle
                  1005+4
                           * Bus de données afficheur
dbat
                           * Registre de statut de la carte
                eid005+6
stat
                eid005+7
                           * Registre de contrôle de la carte
reg ctrl
                EID005+5
reg clavi
     Table de définition des paramètres de l'afficheur
 *******************
          $00,$04,$42
                          * Graphic Hom Adress et Command
          $10,$00,$43
                           * Graphic Area et Command
          $00,$00,$40
                          * Text Hom Adress et Command
          $10,$00,$41
                           * Text Area et Command
ModSet
                           * OR Graphique ou Texte
          equ
                $80
          equ
Pointeur
                $24
                           * Commande pointeur
          equ $94
DispMod
                           * Affichage Texte et ou Graphique
                           * Auto incrémentation pixel ou caractère
AutoInc
          equ $C0
```



```
init acces equ
                 $EF
                      * Pour accéder au bus de données du lcd avec fs=0
                 $E9
                      * cd=1, ce=0, wr=0, rd=1, fs=0 1110 1001
Wrc
           equ
                      * write commande
                     * cd=1, ce=0, wr=1, rd=0, fs=0 1110 1010
rdc
                 $EA
           equ
                      * read commande (Status)
                     * cd=0, ce=0, wr=0, rd=1, fs=0 1110 0001 write data
wrd
                 $E1
                     * cd=0, ce=0, wr=1, rd=0, fs=0 1110 0010 read data
                 $E2
rdd
           equ
**
     Une chaîne de caractère se termine par le caractère '$'
     Attention les espaces dans le texte servent à afficher sans couper
* *
Texte dc.b 'EID005 CLAVIER AFFICHEUR
                                           HORLOGE TEMPS
                      code
     section
 *******
     PROGRAMME PRINCIPALE
                 init_aff * Initialisation
     bsr
     bsr
                lcd cls
                            * Effacement éci
* Envoi Texte : x=2, y=1 : dans D0
     move.b
                 #$20,D0
                            * LSBy
                                     TH
                                                  colonne 1
                            * MSByt
     clr.b
                D1
                 lcd_gotoxy
     bsr
                                        ment
                                              er caractère
     move.1
                 #Texte,A0
                 lcd_out_st
     bsr
                              Sous-progra
                                           e envoi de chaîne
                              de cara
                                       ère
     JMP
                    ITEUR
                                     our au Moniteur
                               =========*
     FIN E
                    MMES
                 #init acces,ctrlaff
busy move.b
                #rdc,ctrlaff
     move.b
     bsr
                 delay
                 dbaff,D4
                                 * Lecture data bus aff
     move.b
                                  * Isoler ST0 STA1 (Status)
     and.b
                 #03,D4
                                  * Si lcd pas prêt
     cmp.b
                #03,D4
                                  * attendre
                busy
     bne
     move.b
                #init_acces,ctrlaff
     rts
```



```
wrcomm : Ecriture commande placée dans D0 ***
****************
wr_comm
                busy
                 #init acces,ctrlaff
                                        * fs = 0; cd, ce, wr et rd = 1
     move.b
                                        * Déposer la commande sur le bus
     move.b
                D0,dbaff
                                        * de donnée
                                        * Générer les impulsions
     move.b
                #wrc,ctrlaff
                                        * d'écriture d'une commande
     bsr
                delay
     move.b
                #init acces,ctrlaff
      wrdata : Ecriture data placée dans D0 ***
wr_data
     bsr
                busy
     move.b
                 #init_acces,ctrlaff
                                                             et^{rd} = 1
     move.b
                D0,dbaff
                                                     data
                                                              le bus de
                                          dc
     move.b
                #wrd,ctrlaff
                                          Gér
                                                         Ilsions
                                                     d'une donnée
     bsr
                 delay
     move.b
                 #init_acces,ctrla
     rts
    rddata : Lecture da
                              à placer
rd_data
     bsr
                                        * fs = 0; cd, ce, wr et rd = 1
     move.b
                                        * Générer les impulsions
     move.b
                                        * d'écriture de
                                        * d'une donnée
     bsr
                                        * Lire et mettre la donnée dans D0
     move
                  init acces, ctrlaff
     move.b
                    itialisation afficheur
* Initial
                de la zone Adresse de début du Texte
init_aff
     clr.b
                 D0
     bsr
                            * LSByte TH = 00
                wr_data
                 wr_data
                            * MSByte TH = 00
     bsr
     {\tt move.b}
                #$40,D0
                            * Commande d'écriture TH
                            * Ecriture TH
     bsr
                wr_comm
                 #$10,D0
                            * LSByte TA = $10 nombre de caractères/ligne
     move.b
                 wr_data
     bsr
     clr.b
                D0
                            * MSByte TA = 0
     bsr
                wr_data
     move.b
                #$41,D0
                            * Commande d'écriture TA
                            * Ecriture TA
     bsr
                wr_comm
```



```
Initialisation zone Adresse de début du Graphique
                 D0
     clr.b
                             * LSByte GH = 00
     bsr
                 wr_data
                #04,D0
                             * MSByte GH = 00
     move.b
                wr data
                             * Ecriture GH
     bsr
                #$42,D0
                             * Commande d'écriture TH
     move.b
     bsr
                 wr comm
     move.b
                 #$10,D0
                             * LSByte GA = $10 nombre de caractères/lique
     bsr
                 wr data
                             * MSByte GA = 0
     clr.b
                D0
     bsr
                 wr data
     move.b
                #$43,D0
                             * Commande d'écriture TA
     bsr
                 wr comm
                             * Ecriture commande TH
* Définition Modes
                 #DispMod,D0 * Mise en mode Graph;
     move.b
     bsr
                 wr comm
     move.b
                 #ModSet,D0
     bsr
                 wr_comm
                             * Mise en mode
     rts
      lcd_gotoxy : Init position car
lcd_gotoxy
                 wr_data
                                         doni
                                                LSByte contenue dans D0
     bsr
                                           dan. D0
                 D1,D0
                               MSByte dal
     move.b
                                            pée MSByte contenue dans D0
                 wr_data
                               Ecrire la de
     bsr
                              * Code d
                                        commande du Pointeur dans D0
     move.b
                 #Pointeur,
     bsr
                               Ecriture
                                        de la commande du Pointeur dans D0
                 wr c
     rts
      lcd cls
                             écran ***
                     acem
******
lcd cls
                               Adresse 00,00 dans D0 et D1
     clr.
     clr.
                  cd gotoxy * Positionner sur le premier caractère
                 #2048,D3
                             * Nombre d'octets de la mémoire écran
     clr.b
                    data
sui
     bsr
                 #AutoInc,D0 * Code auto incrémentation dans D0
                            * Incrémenter le pointeur de caractère
                 wr comm
     bsr
                 #0,D0
     rom
     sub
                 #1,D3
                 suite
                             * Continuer d'effacer tout l'écran
     bne
     rts
```





```
lcd_out_str : Envoi chaîne de caractères ***
*************
** Attention : il faut soustraire la valeur $20 au code ASCII du caractère
lcd out str
               (A0)+,D0 * Pointeur un caractère
     move.b
               #'$',D0 * Est-ce le caractère de fin ?
     cmp.b
                         * Fin si le caractère est le $
     beq
               fin
               #$20,D0 * A cause du code ASCII décalé voir notice
     sub.b
     bsr
               wr data
     move.b
               #AutoInc,D0
     bsr
               wr comm
    bra
               lcd_out_str
fin
     rts
      delay
***
delay move.b
               #$10,D4
bc1
     sub.b
               #1,D4
     bne
               bc1
     rts
tempo move.1
               #$4000,D5
temp1 sub.1
               #1,D5
     bne
                temp1
     rts
     end
```



EID005_TP 2 ECRITURE D'UNE CHAINE DE CARACTERES EN MODE TEXTE SUR L'AFFICHEUR

2.1 Enoncé du sujet

Objectifs :	Etre capable d'utiliser les utilitaires rangés en bible thèque, permettant la gestion de l'afficheur LCB per phique 1, 8x64 pixels.
Cahier des charges :	Ecrire un programme, a a sem leur et en C qui réalisera l'affichage d'une le aîne le cara ères en mode texte. La longueur maxima, de le chaîne est de 128 caractères. On donné a les coorde nées du 1 ^{er} caractère.

Waériel nécessaire :

Micro catinateur (type PC sous Windows 95 ou ultérieur, Cart mère 16/32 les à microcontrôleur 68332, Réf : EID210000 Cart Clavir Che. Horloge Temps réel : EID00500 Câble maison SB, ou à défaut câble RS232, Réf : EGD000003

Alimentation AC C 8V, 1 A Réf : EGD000001,

Documentations nécessaires :

Document : DMS Carte Clavier Afficheur Horloge Temps réel : EID005000 Application Notes for the T6963C LCD Graphics Controller Chip (TOSHIBA) T6963c DOT MATRIX LCD CONTROL LSI (TOSHIBA)

Durée : 1 séance de 3 heures



2.2 Eléments de solution

2.2.1 Description sommaire de l'afficheur

2.2.1.1 Représentation de l'afficheur LCD 128x64

Attention:

Pour des raisons de conformité avec la documentation du constructeur, les variables x et y représentent respectivement l'ordonnée (verticale) et l'abscisse (horizontale). Le point x = 0, y = 0 est en haut à gauche et le point x = 63, y = 127 en bas à droite de l'écran du LCD

Le contrôleur T6963C dispose d'une mémoire de 8 ko.

Mode texte fig.1

Dans l'étude qui suit, la zone texte est placée de l'adresse 0000 à 007 de la mémoire écran (VRAM), soit 128 caractères.

Octet de poids faible : 00 toujours fixe

Octet de poids fort : 00 à 7F en hexadécimal.

Le quartet de poids fort de cet octet désigne le nume o de la colonge y. Le quartet de poids faible désigne le numer de la colonge y.

Exemple

Le caractère numéro 59 est pla é en : x = 3, y = 1; ce qui donne en hexadécimal les valeurs : x = 3, y = 1 d'où el mémoire, le octets suivants pour le paramètre TH (Text Home Adress :

Adress lower = 0x00 (59 en decimal)

MODE TEXTE

				•												,
Conte		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127

fig.1



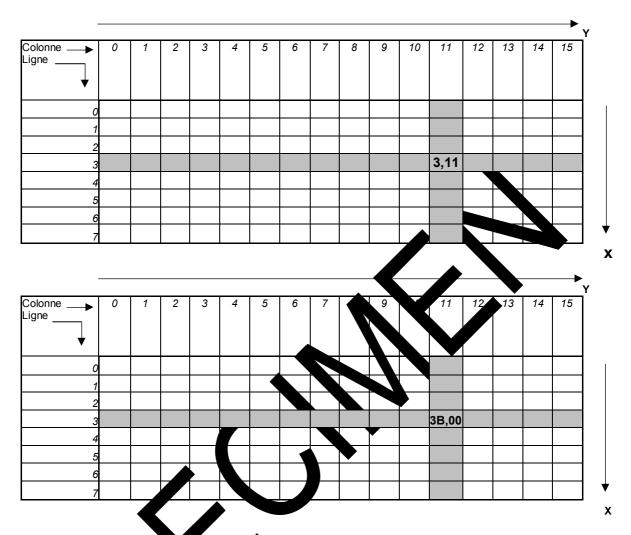


fig.1 suite

Générateur : caracteres inte ¿ (CG ROM)

Le géneraleur de gractères interne utilise un code ASCII décalé de 0x20 ; exemple : la le re A codée 0x Len ASCII est représentée par la valeur 0x21 dans le T6963C (voir ablance ci-c ssous).

Cela signifie que our envoyer un caractère ASCII, il faut soustraire de son code, la valeur 0x20

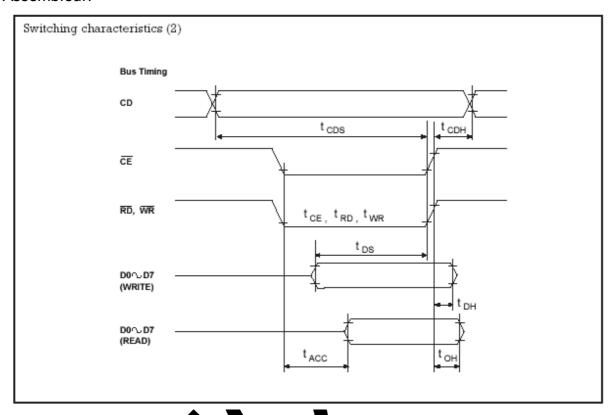
↓ <u>MSB_LSB</u> →	0	1	2	3	4	5	6	7	8	9	A	В	C	D	E	F
0	blank	!	66	#	\$	%	&	-	()	*	+	,	-		-/-
1	0	1	2	3	4	5	6	7	8	9	:	;	٧	П	۸	?
2	(a)	Α	В	С	D	Е	F	G	Н	I	J	K	L	Μ	Ν	О
3	P	Q	R	S	T	U	V	W	X	Y	Z]	_]	<	_
4	\	a	b	С	d	e	f	g	h	i	j	k	1	m	n	0
5	р	q	r	S	t	u	v	w	х	У	z	{		}	?	blank
6	Ç	ü	é	â	ä	à	å	ç	ê	ë	è	ï	î	ì	Ä	Å
7	É			ô	ö	ò	û	ù	ÿ	ö	ü	¢	£	¥		f





Chronogrammes Ecriture / Lecture de données ou commandes

Ces chronogrammes doivent être générés pour chaque accès au LCD. Ils sont réalisés et décrits en détail notamment dans les sous-programmes en Assembleur.



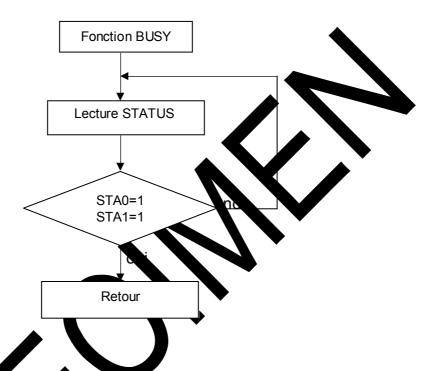
La génération de cha un de control de character de correspondant.



2.2.1.2 Gestion de l'afficheur LCD

Avant chaque mouvement (écriture ou lecture) donnée (commande ou donnée) entre l'afficheur et le processeur de commande, il faut s'assurer que le LCD est prêt à exécuter l'opération.

D'où la nécessité de commencer par tester les bits d'état (ou de statut) de son registre d'état : STA0 et STA1 ; ce qui donne l'organigramme suivant :



2.2.1.3 Instruct de de l'accheur

Les instruction des ffich y sont onnées dans le tableau ci-dessous.

Les organigne mes d'dessous définissent les modes d'écriture d'instruction de commande nécessant 2, 1 ou 0 octets.

Les rganiers reservivants définissent en détail les modes d'écriture et de lecture des consaindes et des données.



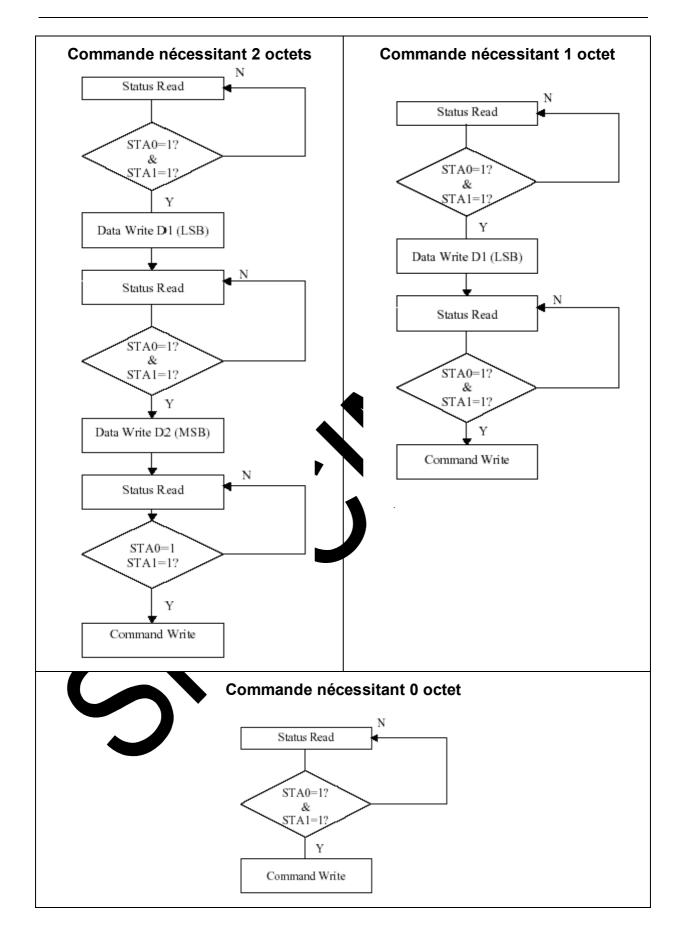
T6963C Instruction Set

Commands	D7	D6	D5	D4	D3	D2	D1	D0	Description	Execute Time
Pointer Set	0	0	1	0	0	N2	N1	N0		Status
						0	0	1	Cursor Pointer Set	check
						0	1	0	Offset Register Set	1
						1	0	0	Address Pointer Set	
Control Word	0	1	0	0	0	0	N1	N0		32 x 1/fosc
Set Commands							0	0	Text Home Address Set	1
							0	1	Text Area Set	1
							1	0	Graphic Home Address Set	1
							1	1	Graphic Area Set	1
Mode Set	1	0	0	0	CG	N2	N1	N0		32 x
					0		\vdash		CG ROM Mode	1/fose
	\vdash		\vdash		1		\vdash		CG RAM Mode	1
	\vdash				<u> </u>	0	0	0	"OR" Mode	1
						0	0	1	"EXOR" Mode	1
						0	1	1	"AND" Mode	1
	ш					1	0	0	Text only (attribute capability)	1
Display Modes	1	0	0	1	N3	N2	N1	NO	, , , , , , , , , , , , , , , , , ,	32 x
	Ė	_	_	<u> </u>	0		<u> </u>		C	1/fosc
	\vdash	_	\vdash		1		 	\vdash	Graphics Off	-
	\vdash	_			1	0	 		Graphics On Text Off	1
					_	1	├		Text On	-
	-	\vdash	\vdash		\vdash	١,	0		Cursor Off	1
	\vdash		\vdash		\vdash		1		Cursor On	1
	\vdash		\vdash		\vdash		<u> </u>	0	Cursor On Cursor blink Off	-
	-	-	\vdash		\vdash		\vdash	1	Cursor blink On	1
Cursor Pattern	1	0	1	0	0	N2	N1	N0	N2-N0: No. of lines for cursor +1	32 x
Cursor Pattern	1	u u	1	u u	U	NZ	NI	190	N2-N0: No. of lines for cursor +1	1/fosc
Select						0	0	0	Bottom Line cursor]
						0	0	1	2 line cursor]
	ldot]
						1	1	1	8 line cursor (block cursor)	
Data Auto	1	1	0	0	0	0	N1	N0		32 x 1/fosc
Read/Write	-				-		0	0	Data Auto Write Set	1
	-						0	1	Data Auto Read Set	1
							1	0	Auto reset (Address pointer auto-	1
									incremented) for continuous rd/wr	
Data Read/Write	1	1	0	0	0	N2	N1	N0		
						0			Address Pointer up/down]
						1			Address Pointer unchanged]
							0		Address Pointer up]
							1		Address Pointer down	1
								0	Data Write	1
								1	Data Read	
Screen Peeking	1	1	1	0	0	0	0	0	Read Displayed Data	Status
Screen Copy	1	1	1	0	1	0	0	0	Copies 1 line of displayed data whose	Status
(Note 3)							l		address is indicated by the Address	check
	⊢		<u> </u>				—	_	Pointer to Graphic RAM area	—
Bit Set/Reset	1	1	1	1	N3	N2	N1	N0	N2-N0 indicates the bit in the pointed	Status
	\vdash	_							address	check
	\vdash	_	_	_	0		<u> </u>	_	Bit Reset	-
	\vdash	_	—		1	_	_		Bit Set	1
	\vdash	_	<u> </u>	_	<u> </u>	0	0	0	Bit 0 (LSB)	1
	\vdash	_			_	0	0	1	Bit 1	-
	\vdash	_	_	_		1	+	\vdash	Dr. 7 (Men.	-
						1	1	1	Bit 7 (MSB)	

Note:

1. * = DONT CARE

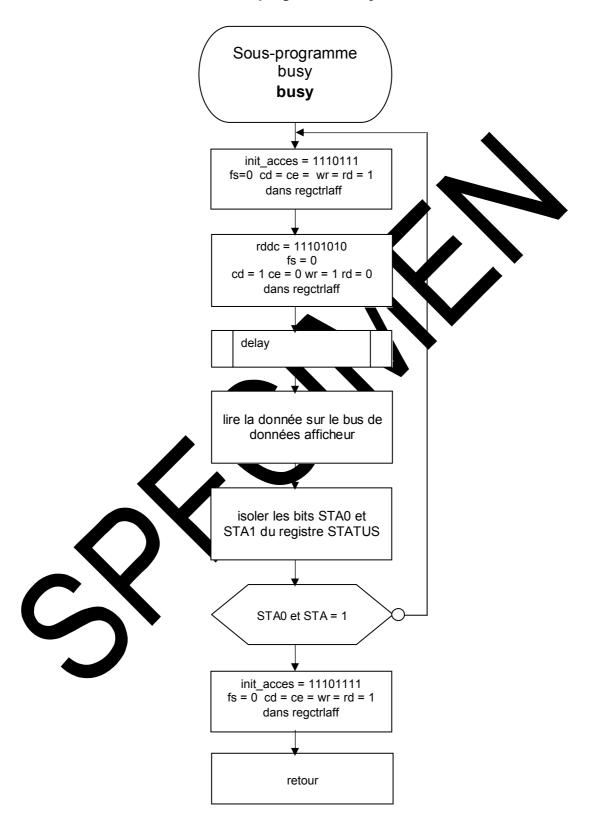






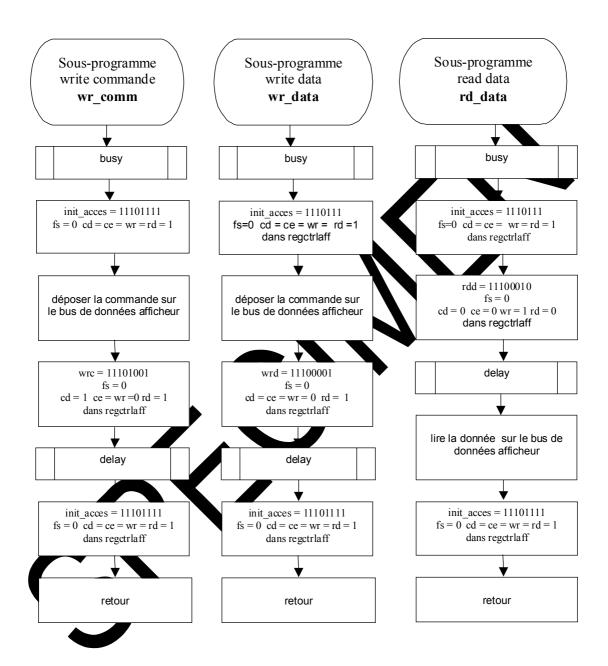
Organigrammes détaillés des sous-programmes principaux

Sous-programme busy





Sous-programmes écriture lecture







2.2.1.4 Organigramme principal

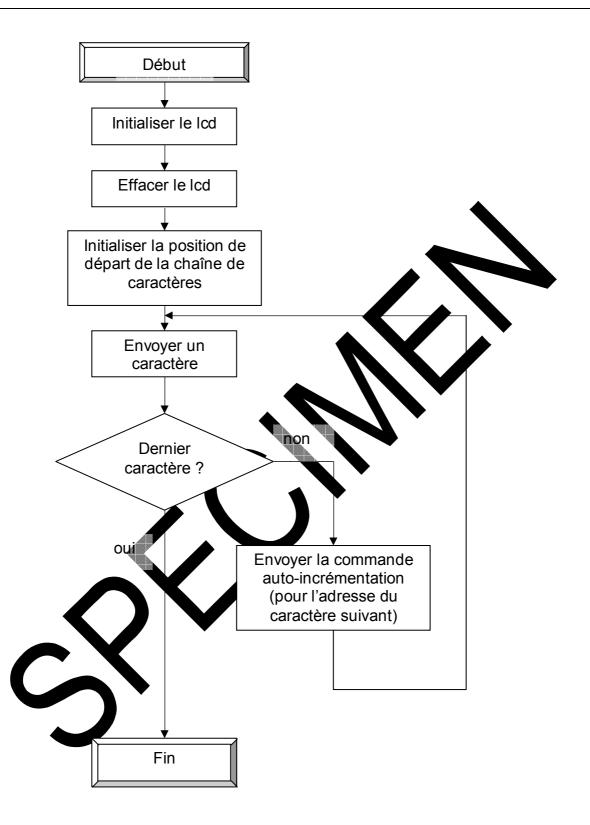
Vous disposez des utilitaires suivants avec leurs commentaires ; tous ces utilitaires font déjà appel au test de STA0 et STA1

Utilitaires	Commentaires
<pre>void init_aff()</pre>	Initialisation des
	paramètres TH, TA, GH
	GA Mode
<pre>void lcd_cls()</pre>	Effacement de l'écran
<pre>void lcd_write_command(unsigned char commande)</pre>	Ecriture 'une
	commande
<pre>void lcd_write_data(unsigned char data)</pre>	Ecriture d't donnée
<pre>void lcd_gotoxy(unsigned char px, unsigned char py)</pre>	Dél de
	caracte e ou d'un
	pixel ;
	px t py s t les
	nnées lower et
	apper.

Important:

Pour utiliser ses fonctions ainst que le strier afficheur avec le compilateur C/C++, il faut, dans le mou du ogiciel eid210, aller dans le menu configuration puis gnv 2/C++. Ensure aver dans l'onglet *linker*, cliquer sur *ajouter*, sélectionner le fittier « EID005_Lib.o » et finir en cliquant sur ouvrir.







2.2.2Programme en C

```
*********
     TP SUR LA CARTE EID005
    *********
     Ecrire un programme en Assembleur
     et en C et qui réalise l'affichage
     d'une chaîne de caractères.
     longueur maximale de la chaîne : 128 caractères *
   *************
     Nom du FICHIER : EID005_TP2.c
     Inclusion des fichiers de définition
#include "eid005.h"
#include <stdio.h>
#include <string.h>
#include <math.h>
//********
// DECLARATIONS
//********
unsigned char Texte [] = { "E
                                                     AFFICHEUR,
             REEL $"};
HORLOGE TEMPS
// Les espaces dan le t
                      te
                            mettent
                                    éviter de couper un mot
//========
// FONCTION PRI
                  PALE
//=======
main()
int i ;
                               Variable de comptage des caractères
                          //
init
                               Initialisation de l'afficheur
                          //
lcd
                               Effacement de l'afficheur
                          //
                               Définition de la position de départ de
lcd
    otoxy
                          //
la
     Bougle d
             comptage du nombre de caractères
for (i=0; Texte [i]!= 0x24; i++) // Fin de comptage par la détection du
caractère "$"
lcd write data((unsigned char) (Texte[i]-0x20));
caractère
lcd write command(0xC0);  // Envoyer la commande d'incrémentation
d'adresse
//
     Fin du programme
```



2.2.3 Programme en Assembleur

```
TP SUR LA CARTE EID005
    ********
     Ecrire un programme en Assembleur
     et en C et qui réalise la lecture d'un touche
     sur un clavier matricé 4 x 4, en mode pooling
    **************
     Nom du FICHIER : EID005 TP2.src
     ******
***********
     La commande ou la donnée à écrire sont d'abord rap
* *
     La donnée lue est rangée dans D0
* *
     La position x,y d'un caractère est placée resp
                                                          dans
     L'adresse de début d'une chaîne de caractère
                                                          ans A0
**********
     Bits du registre de contrôle de l'affi
     ctrlaff : b0=rd
                b1=wr
                b2=ce
                b3=cd
* Inclusion du fichier défini
* de l'EID210
     include EID210.de
     section var
                         es physiques de l'afficheur et du clavier
     Définition des
eid005
                   000
                             Adresse de base Carte clavier_afficheur_rtc
ctrlaff
                   005+
                             Registre contrôle afficheur : bus de
                           * contrôle
                 id005+4
dbaff
                           * Bus de données afficheur
                eid005+6
                           * Registre de status de la carte
stat
          equ
     trl equ
                  d005+7
                           * Registre de contrôle de la carte
reg
                           * reg_clavier
                                          equ EID005+5
     Table de
               éfinition des paramètres de l'afficheur
           $00,$04,$42
                           * Graphic Hom Adress et Command
           $10,$00,$43
                           * Graphic Area et Command
           $00,$00,$40
                           * Text Hom Adress et Command
           $10,$00,$41
                           * Text Area et Command
          equ
                           * OR Graphique ou Texte
ModSet
                $80
          equ
                           * Commande pointeur
Pointeur
                $24
          equ
DispMod
                $94
                           * Affichage Texte et ou Graphique
AutoInc
          equ
                $C0
                           * Auto incrémentation pixel ou caractère
```



```
* Pour accéder au bus de données du lcd avec fs=0
init_acces equ
               $EF
               $E9
                   * cd=1, ce=0, wr=0, rd=1, fs=0 1110 1001
Wrc
          equ
                    *write commande
               $EA
                   * cd=1, ce=0, wr=1, rd=0, fs=0 1110 1010
rdc
          equ
                    *read commande (Status)
               $E1
                    * cd=0, ce=0, wr=0, rd=1, fs=0 1110 0001 write data
wrd
          equ
                   * cd=0, ce=0, wr=1, rd=0, fs=0 1110 0010 reda data
rdd
               $E2
          equ
    Une chaîne de caractère se termine par le caractère '$'
    Attention les espaces dans le texte servent à un afficher sans couper
    les mots
Texte dc.b 'EID005 CLAVIER AFFICHEUR HORLOGE TEMPS
                                                     REEI
     section
                    code
********
    PROGRAMME PRINCIPALE
                        * Initialisation
     bsr
               init aff
     bsr
               lcd cls
                          * Efface
                                      écrar
* Envoi Texte : x=2, y=1 : dans D0
                                          e 0 colonne 1
    move.b
               #$20,D0
    clr.b
               D1
                           MSByte TH
    bsr
               lcd_gotoxy
                          * Positionnement ler caractère
     move.1
               #Texte_A0
     lcd out str :
                    voi cha
                              de g
                                   actères1cd out str ***
** Attention il
                      raire 🖎 valeur $20 au code ASCII du caractère
               aut`
1cd out st:
                          * Pointeur un caractère
     move
                  ) + , D0
                 $',D0
     cmp.b
                         * Est-ce le caractère de fin ?
               fin
                          * Fin si le caractère est le $
     sub.b
                $20,D0
                          * A cause du code ASCII décalé voir notice
     bsr
                  data
               #AutoInc, D0
     bsr
               wr comm
               1cd out str
JMP
               MONITEUR
                              Retour au Moniteur
FIN PROGRAMME PRINCIPAL *
*********
```



```
LES SOUS-PROGRAMMES
********
      Busy ***
*****
                #init acces,ctrlaff
busy move.b
               #rdc,ctrlaff
     move.b
     bsr
                delay
                                  * Lecture data bus aff
     move.b
               dbaff,D4
                                 * Isoler ST0 STA1 (Status)
     and.b
               #03,D4
                                  * Si lcd pas prêt
     cmp.b
                #03,D4
                                 * attendre
     bne
                busy
     move.b
                #init acces,ctrlaff
     rts
      wrcomm : Ecriture commande placée dans DQ
wr_comm
     bsr
                busy
     move.b
                #init_acces,ctrlaff
                                        * fs
                                                         wr et rd = 1
                                         Dépo
                                                    commande sur le bus
                D0,dbaff
     move.b
                                                ée
     move.b
                #wrc,ctrlaff
                                                 les impulsions
                                               commande
                 delay
     bsr
     move.b
                            ctrlaff
                 #init_acce
     rts
      wrdata
                 Ec
                                lacée
                                      ans D0 ***
wr data
     bsr
                              rlaff
                                       * fs = 0; cd, ce, wr et rd = 1
     move
                    dbaff
                                        * Déposer la data sur le bus de
                                        * donnée
                                       * Générer les impulsions
                 #wrd,ctrlaff
                            * d'écriture de
                                       * d'une donnée
                 delay
     move.b
                 #init acces,ctrlaff
```



```
rddata : Lecture data à placer dans D0 ***
rd_data
                busy
     bsr
                                        * fs = 0; cd, ce, wr et rd = 1
     move.b
                 #init acces,ctrlaff
                #rdd,ctrlaff
                                        * Générer les impulsions
     move.b
                                        * d'écriture de
                                        * d'une donnée
                delay
     bsr
                dbaff,D0
                                  * Lire et la mettre la donnée dans D0
     move.b
                #init acces,ctrlaff
     move.b
     rts
      init aff : Initialisation afficheur
init_aff
* Initialisation de la zone Adresse de début
     clr.b
                D0
                wr data
                             * LSByte \underline{T}H = 00
     bsr
                wr_data
     bsr
                             * MSByte
                                          00
                             * Command
     move.b
                #$40,D0
                             * Ecr
     bsr
                wr_comm
                              LSByt'
                                                  re de caractères/ligne
     move.b
                 #$10,D0
     bsr
                wr_data
     clr.b
                D0
                              MSByte TA
     bsr
                 wr_data
     move.b
                 #$41,D0
                              Command
                                       d'écriture TA
                                        TA
     bsr
                 wr_co
                               Ecriture
* Initialisation de
                   a zone
                            Adres
                                     début du Graphique
                 D0
     clr.b
                    da
                                SByte GH = 00
     bsr
                              MSByte GH = 00
     move
                    ,D0
                              Ecriture GH
     bsr
                    data
                  $42,D0
                             * Commande d'écriture TH
     move.b
                 wr comm
     move.b
                  $10,D0
                             * LSByte GA = $10 nombre de caractères/ligne
     bsr
                   data
                 D0
                             * MSByte GA = 0
                 wr data
     bsr
                 #$43,D0
                            * Commande d'écriture TA
     mor
                 wr_comm
                             * Ecriture commande TH
     bsr
* Définition Modes
     move.b
                 #DispMod,D0 * Mise en mode Graphique et/ou Texte
     bsr
                wr_comm
     move.b
                #ModSet,D0
                 wr_comm
                            * Mise en mode Graphique OU Texte
     bsr
     rts
```



```
lcd_gotoxy : Init position caractère
*************
lcd_gotoxy
              bsr
              D1,D0 * MSByte dans dans D0 wr_data * Ecrire la donnée MSByte contenue dans D0
     move.b
     bsr
             #Pointeur, D0 * Code de commande du Pointeur dans D0
     move.b
     bsr
              wr comm * Ecriture de la commande du Pointeur dans D0
     rts
     lcd cls : Effacement écran ***
*********
1cd cls
                         * Adresse 00,00 dans D0 et
     clr.b
              D0
     clr.b
     bsr
              lcd gotoxy * Positionner sur le
             #2048,D3
                         * Nombre d'octets d
                                            a mémo
     move.w
     clr.b
suite bsr
             wr data
             #AutoInc,D0 * Code auto ind
     move.b
                       * Incrémenter le
     bsr
             wr comm
             #0,D0
     move.b
     sub.w
              #1,D3
     bne
               suite
                         * Continu
                                                l'écran
     rts
     delay
delay move.b
               #$10,D4
bc1
     sub.b
               #1,D4
     bne
               bc1
     rts
tempo move.1
temp1 sub.
                 D5
     bne
     rts
```





EID005_TP 3 LECTURE D'UN CLAVIER MATRICE EN MODE POLLING

3.1 Enoncé du sujet

Objectifs :	Etre capable d'utiliser les utilitaires rangés en hibliothèque, permettant la gestion d'un clavier matricé 16 pouts (4x4).
Cahier des charges :	Ecrire un programme en asse abour et . C qui réalisera la lecture d'une touche la slavier nativé en mode pooling. Cette lecture sedera par des stius de l'activation d'une touche appuyée.

Matériel nécessaire :

Micro ordina eur de tyle PC 3 2 Windows 95 ou ultérieur, Carte mère 1632 by à microcontrôleur 68332, Réf : EID210000 Carte care r Afric eur Horloge Temps réel : EID00500

Câb de liaison US ou à défaut câble RS232, Réf : EGD000003

Alimentation Ac. 1C 8.7, 1 A Réf : EGD000001,

Documentations nécessaires :

Document: DMS Carte Clavier Afficheur Horloge Temps réel: EID005000 Application Notes for the T6963C LCD Graphics Controller Chip (TOSHIBA) T6963c DOT MATRIX LCD CONTROL LSI (TOSHIBA)

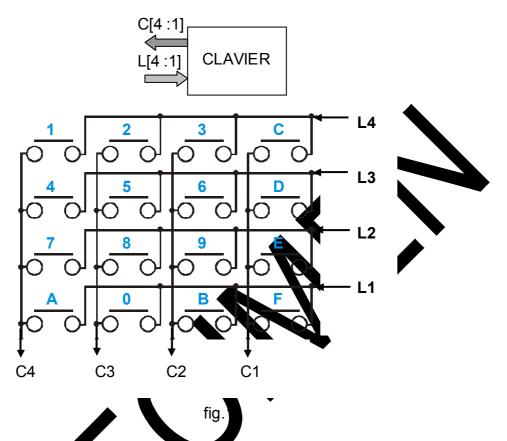
Durée : 1 séance de 3 heures



3.2 Eléments de solution

3.2.1 Description sommaire du clavier

3.2.1.1 Représentation du clavier 4x4 fig.1



Les 4 lignes de l'avier sont cas sur la sortie de l'EPLD de gestion de l'EID005.

Chaque sortice L'EP L'est mur d'une résistance de pull-up (tirage à Vcc).

Les 4 Jonna sont câbées sur des entrées.

3.2.12 Princip général de lecture d'un clavier matricé

Lorsque le sorties commandant les 4 lignes sont au niveau logique 1, les 4 bits correspondant aux 4 colonnes sont à 1 quelque soit le nombre de touches appuyées.

Le principe de la lecture par la méthode pooling consiste à fixer 1 ligne L_j 0, les 3 autres à 1 (balayage lignes) puis à détecter le numéro de la colonne C_i à 0 (balayage colonnes).

L'intersection $L_{j,x}C_i$ donne le caractère correspondant.

Il ne reste plus qu'à coder ce caractère pour lui donner la valeur binaire ou hexadécimale voulue.





3.2.1.3 Lecture d'une touche appuyée en mode pooling

Pour faire cette lecture il faut

- a) balayer les lignes en positionnant une ligne n° i à 0
- b) lire les colonnes
- c) tester si une colonne est à 0 (au moins 1 touche appuyée)
- d) déterminer le poids de la colonne
- e) extraire la valeur de la touche
- f) afficher la valeur de la touche à l'écran du PC.
- g) tester toutes les touches.

3.2.1.4 Organigramme principal

Le clavier est codé par une matrice 4x4 en hexadéc dal comple indiqué cidessous.

Attention à la position des lignes dans le le leau de cod re par rapport à celle de la matrice elle-même.

Poids de la colonne				4
	8		2	1
C ₁				
L _J				0
	C_4	C ₃	C_2	C ₁
	0-	02	03	0C
L;	4	05	06	0D
	07	80	09	0E
	0A	00	0B	0F

Les lignes et ennes sont décrites dans le fichier eid005.h de la carte



La portion correspondante est ci-dessous.

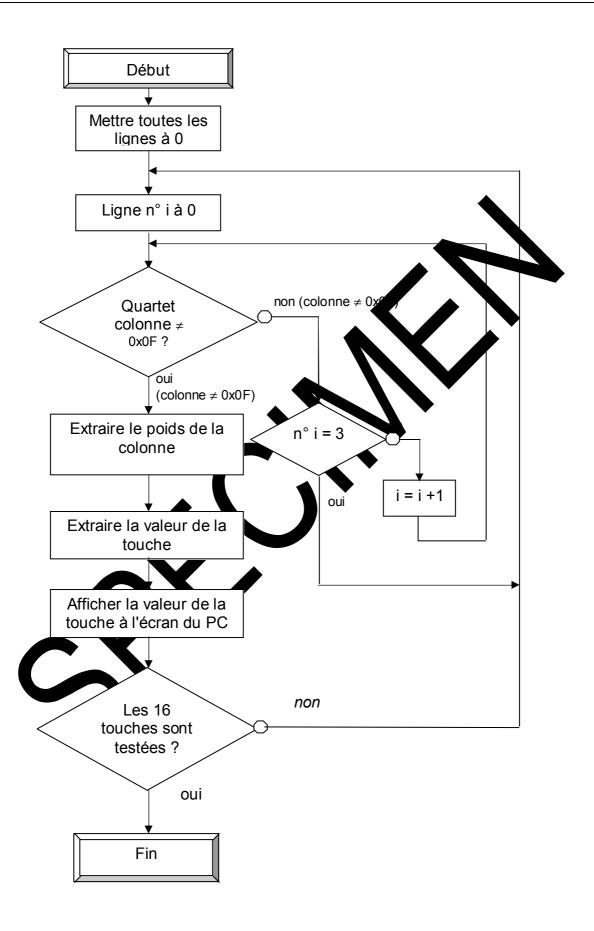
Une ligne i est définie par la variable :

- clavier.matrice.ligne = i ; (0 =< i =< 2</p>
 Les colonnes sont lues à travers la variable.
 - > clavier.matrice.colonne.

Important:

Pour utiliser le clavier affirmeur avec le consilateur C/C++, il faut, dans le menu du logiciel eid210 aller dans le menu configuration puis gnu C/C++. Ensuite aller dans l'onglet vilinger*, cliquer sur *ajouter*, sélectionner le fichier EI 05_Lib.o et finir en cliquant sur ouvrir.







3.2.2 Programme en C

```
/************
     TP SUR LA CARTE EID005
    ***********
     Ecrire un programme en Assembleur
     et en C et qui réalise la lecture d'un touche
     sur un clavier matricé 4 x 4, en mode pooling *
     Nom du FICHIER : EID005 TP3.c
    Inclusion des fichiers de définition
#include "eid005.h"
#include <stdio.h>
#include "eid210.h"
//=============
// FONCTION PRINCIPALE
//=============
main ()
short
        Touche[4][4] = \{ 0x0A, 0x0 \}
                       0x07, 0x08
                        0x04
short V, ValeurTouche;
int tmp ;
int i, j, k, m;
                   ces leg
                                    e la matrice à 0
    Positionner t
                           // Element ligne (4 bits) de la structure
clavier.matrice.li
appartenant
                              la variable clavier de l'union
reg_clavier
j=0;
do
{j++;
               ur une touche \n");
      ("Appuyer
     Touch
             ppuy
while (clavier natrice.colonne & 0x0F) != 0x0F)
                                                            // Attente
appui touche
     for (i = 0; i < 4;)
          clavier.matrice.ligne = ~(1 << i); // Toutes les colonnes à 1</pre>
           tmp = i;
          if (touche == 1) break;
                                          // touche = 1 ==> au moins
une touche appuyée
           i++ ;
           for (k = 0 ; k<10000 ; k++); // temporisation
    Extraire le poids de la colonne
```





```
clavier.matrice.colonne = Elément colonne (4 bits) de la structure
//
      appartenant à la variable clavier de l'union reg_clavier
//
      Puis estraire la valeur de la touche appuyée
      Extraire la valeur de la touche appuyée
      clavier.matrice.colonne & 0xF;
switch((~(clavier.matrice.colonne)) & 0x0F )
      case 1 : ValeurTouche= Touche [tmp][3]; break; //Colonne poids 1
      case 2 : ValeurTouche= Touche [tmp][2]; break; //Colonne poids 2
      case 4 : ValeurTouche= Touche [tmp][1]; break; //Colonne poids 4
      case 8 : ValeurTouche= Touche [tmp][0]; break; //Colonne poids 8
}
     Afficher loa valeur de la touche à l'écran du PC
printf ("La touche appuyee est : %x\n\n", ValeurTouche )
while (touche)
                              // Attente relâche toug
            for (k = 0 ; k<1000000 ; k++);
                                                    empori
     Toutes les touches sont testées ?
while (j<17);
}
//
      Fin du programme principal
```



3.2.3 Programme en Assembleur

```
TP SUR LA CARTE EID005
     Ecrire un programme en Assembleur
     et en C et qui réalise la lecture d'un touche
     sur un clavier matricé 4 x 4, en mode pooling
    *************
     Nom du FICHIER : EID005_TP3.src
     *****
     La commande ou la donnée à écrire sont d'aborg
     La donnée lue est rangée dans D0
     La position x,y d'un caractère est placée
                                               specti
                                                              ns DO et
D1
**
                                                    placée
     L'adresse de début d'une chaîne de car
*********
     Bits du registre de contrôle de
     ctrlaff :
                b0=rd
                b1=wr
                 b2=ce
                 b3=cd
                 b4=fs
                                       fférents labels
* Inclusion du fichier
                             ant les
* de l'EID210
     include EID2
                    def
     section var
     Défini
                    adr
                              hysiques de l'afficheur et du clavier
eid005
                   30000
                            * Adresse de base Carte clavier_afficheur_rtc
ctrlag
                 eid005+3
                            * Registre contrôle afficheur : bus de
                            * contrôle
                   d005+4
                                 * Bus de données afficheur
dbat
                 e1d005+6
                            * Registre de status de la carte
stat
                 eid005+7
                            * Registre de contrôle de la carte
reg ct
           equ
                 EID005+5
reg clavier eg
     Table de définition des paramètres de l'afficheur
TabDef
           dc.b $00,$04,$42
                                 * GH
           dc.b $10,$00,$43
                                  * GA
                                 * TH
           dc.b $00,$00,$40
           dc.b $10,$00,$41
                                 * TA
ModSet
           equ $80
                                 * OR Graphique ou Texte
Pointeur
                 $24
                                 * Commande pointeur
           equ
DispMod
                 $94
                                 * Affichage Texte et ou Graphique
           equ
                 $C0
                                  * Auto incrémentation pixel ou caractère
AutoInc
           equ
```



```
$EF
                       * Pour accéder au bus de données du lcd avec fs=0
init_acces equ
                  $E9
                       * cd=1, ce=0, wr=0, rd=1, fs=0 1110 1001
wrc
            equ
                        * write commande
                       * cd=1, ce=0, wr=1, rd=0, fs=0 1110 1010
rdc
                  $EA
            equ
                       * read commande (Status)
                       * cd=0, ce=0, wr=0, rd=1, fs=0 1110 0001
                  $E1
wrd
            equ
                        * write data
                  $E2
                        * cd=0, ce=0, wr=1, rd=0, fs=0 1110 0010
rdd
            equ
                        * reda data
            dc.b 'EDD-----005 $ '
Texte1
            dc.b '?!$'
Texte2
Texte3
            dc.b 'FIN $4
      Tableau des valeurs des touches du clavier
Colonne0
                  dc.b $0C,$0D,$0E,$0F
Colonne1
                  dc.b $03,$06,$09,$0B
Colonne2
                  dc.b $02,$05,$08,$00
Colonne3
                  dc.b $01,$04,$07,$0A
      Tableau des coordonnées d'affichage des
                  dc.b $1A,$3A,$5A,$7A
                  dc.b $18,$38,$58
                  dc.b $16,$36,$56,$
                  dc.b $14,$3
      section
                        code
      PROGRAMME PRING
                                  itialisation Afficheur
      bsr
      bsr
                                 ffacement écran
* Envoi Te
                      y=0
                   $00,D0
                              * LSByte TH : ligne 0 colonne 1
      move.b
                  D1
                              * MSByte TH
      bsr
                   cd gotoxy
      mov
                     xte1,A0
                  1cd out str
                ut affichage touches clavier
* Définition
      move.1
                  #15,D7
      clr.b
                  D1
```



```
* Lecture du clavier
test clav
    bsr
               Touch
     bsr
              Val_Touch * La valeur de la touche est dans D0
              #$0A,D0 * Conversion Hexa --> ASCII décalé de $20,
     cmp.b
                         * voir Générateur de caractères T6963C :
              sup10
     bcc
              #$10,D0 * add $10 si code < $0A
     add.b
              envoi
    bra
              #$17,D0 * si non add $17
sup10 add.b
envoi bsr
              wr data
     move.b #$C0,D0
     bsr
              wr comm
     dbeq
              D7,test clav
* Envoi Texte2 : x=7, y=0
                          * LSByte TH : ligne 7
     move.b
             #$70,D0
     clr.b
              D1
                          * MSByte TH
              lcd gotoxy
             #Texte2,A0
     move.1
               lcd_out_str
     bsr
* Envoi Texte3 : x=7, y=d
                                                  nne 13
     move.b
               #$7d,D0
                          * LSByte
     clr.b
               D1
                          * MSB
     bsr
               lcd_gotoxy
     move.1
             #Texte3,A0
     bsr
               lcd_out_st;
*==============
     JMP
                                    r au Moniteur
               MONI
                               Reto
*===========
                                    <u>'</u>=========*
     FIN PROGRA
        SO
                 RAMMES
     SOUS
                  GESTION DE L'AFFICHEUR
               #init_acces,ctrlaff
busy move.b
               #rdc,ctrlaff
     move.b
     bsr
               delay
     move.b
               dbaff,D4
                            * Lecture data bus aff
                               * Isoler ST0 STA1 (Status)
     and.b
               #03,D4
               #03,D4
     cmp.b
                               * Si lcd pas prêt
               busy
                               * attendre
     bne
               #init_acces,ctrlaff
     move.b
     rts
```



```
wrcomm : Ecriture commande placée dans D0 ***
**************
wr_comm
                busy
     bsr
                #init_acces,ctrlaff
                                       * fs = 0; cd, ce, wr et rd = 1
     move.b
     move.b
                D0,dbaff
                                       * Déposer la commande sur le bus
                                        * de donnée
                                       * Générer les impulsions
                #wrc,ctrlaff
     move.b
                                       *d'écriture de
                                        * d'une commande
                delay
     bsr
                #init acces,ctrlaff
     move.b
     rts
      wrdata : Ecriture data placée dans D0 ***
wr_data
                busy
     move.b
                #init acces,ctrlaff
     move.b
                D0,dbaff
                                                             le bus de
                                         dc
     move.b
                #wrd,ctrlaff
                                                         1sions
                                          d'é
                                                    de
     bsr
                delay
     move.b
                #init_acces,ctrlaff
     rts
      rddata : Lecture da
                           a à placer dan
rd data
     bsr
                          s,ctrlaff
     move.b
                                        * fs = 0; cd, ce, wr et rd = 1
                                        * Générer les impulsions
     move.b
                                        * d'écriture de
                                        * d'une donnée
     bsr
                                        * Lire et mettre la donnée dans D0
     move.b
                  baff,D0
                 #init acces,ctrlaff
     rts
                : Initialisation afficheur
                                            ***
init aff
 Initialisation de la zone Adresse de début du Texte
     clr.b
                D0
                wr_data
     bsr
                            * LSByte TH = 00
     bsr
                wr_data
                            * MSByte TH = 00
     move.b
                #$40,D0
                           * Commande d'écriture TH
                wr_comm
     bsr
                            * Ecriture TH
     move.b
                #$10,D0
                            * LSByte TA = $10 nombre de caractères/ligne
     bsr
                wr_data
     clr.b
                D0
                            * MSByte TA = 0
     bsr
                wr data
```



```
#$41,D0
                             * Commande d'écriture TA
     move.b
                 wr_comm
                             * Ecriture TA
     bsr
* Initialisation de la zone Adresse de début du Graphique
                 D0
     clr.b
                            * LSByte GH = 00
     bsr
                wr data
                            * MSByte GH = 00
               #04,D0
     move.b
                wr data
                            * Ecriture GH
     bsr
                            * Commande d'écriture TH
     move.b
                #$42,D0
     bsr
                wr comm
                #$10,D0
                             * LSByte GA = $10 nombre de caractères/ligne
     move.b
     bsr
                wr data
                             * MSByte GA = 0
     clr.b
                D0
     bsr
                wr data
     move.b
                #$43,D0
                             * Commande d'écriture TA
     bsr
                 wr comm
                             * Ecriture commande TH
* Définition Modes
                 #DispMod,D0 * Mise en mode
     move.b
     bsr
                 wr comm
     move.b
                 #ModSet, D0
                             * Mise en_mode G
     bsr
                 wr_comm
     rts
      lcd_gotoxy : Init position ca
lcd_gotoxy
     bsr
                 wr data
                               Ecrire
                                        donnée LSByte contenue dans D0
                                        ns dans D0
     move.b
                 D1,D
                               MSByte o
     bsr
                      ta
                                rire
                                        donnée MSByte contenue dans D0
                                      e commande du Pointeur dans D0
     move.b
                     inteur
     bsr
                              Ecriture de la commande du Pointeur dans DO
                     omn
     rts
      lcd
                    acemen
                            écran ***
                     *****
lcd
     clr.b
                             * Adresse 00,00 dans D0 et D1
     clr
                 lcd gotoxy * Positionner sur le premier caractère
     move.w
                 #2048,D3
                             * Nombre d'octets de la mémoire écran
     clr
                 D0
suite bsr
                 wr data
                 #AutoInc,D0 * Code auto-incrémentation dans D0
     move.b
                 wr_comm * Incrémenter le pointeur de caractère
     bsr
     move.b
                 #0,D0
     sub.w
                 #1,D3
                             * Continuer d'effacer tout l'écran
     bne
                 suite
     rts
```



```
lcd_out_str : Envoi chaîne de caractères ***
*************
** Attention il faut soustraire la valeur $20 au code ASCII du caractère
lcd_out_str
                           * Pointeur un caractère
     move.b
                (A0) + , D0
                #'$',D0
                           * Est-ce le caractère de fin ?
     cmp.b
                           * Fin si le caractère est le $
                fin
     beq
                          * A cause du code ASCII décalé voir notice
                #$20,D0
     sub.b
               wr data
     bsr
               #AutoInc,D0
     move.b
     bsr
                wr comm
     bra
                1cd out str
fin
     rts
***
                ***
     delay
******
delay move.b
                #$10,D4
     sub.b
                #1,D4
bc1
     bne
                bc1
     rts
tempo move.1
                #$4000,D5
temp1 sub.1
                #1,D5
     bne
                temp1
     rts
     Sous programmes Gestio
                            clavier
     Détection d'ung
                      ouche
                                      mode pooling **
    test du regs
                    e clay
                                    lavier
Touch
                                   Ligne 3
     move.b
                                 * N° ligne
     move
                   D2
col i move
                                 * Pointer une colonne
                   reg cl
                           ier
                  eg clavier,D0
     move.b
                                 * Lire les lignes
                #$4000,D4
     bsr
                  empo
                #$0F,D0
     cmp.b
                #$0F,D0
                                 * Touche appuyée pour cette colonne ?
     beg
                balayer
                                 * Pas de touche appuyée pour cette
colonne
                                 * Aller tester si touche lâchée
     bsr
                Lacher
     rts
balayer
     rol.b
                #1,D1
                                 * Ligne suivante
     sub.w
                #1,D2
                                 * n° ligne suivante
     bge
                col i
     bra
                Touch
```



```
* Attendre de relâcher la touche
Lacher
     move.b
              reg_clavier,D3
     bsr
               tempo
     and.b
               #$0F,D3
                                * Isoler le quartet Colonne
               #$0F,D3
                                * Touche non relâchée ?
     cmp.b
                                * Relire les colonnes si touche toujours
     bne
               Lacher
appuyée
                                * Sinon retour de sous prog.
     rts
     Calcul de la valeur de la touche appuyée **
*************
Val Touch
                                * Isoler le n° de la ligne
     and.l
              #$F,D2
              #Colonne0,A0 * Pointer sur la colo
     move.1
              #$0E,D0
                               * Est-ce la colon
     cmp.b
             beq
     move.1
                                               color
     cmp.b
                                              onne
              Val Colonne1
                               * Si oui
     beq
              #Colonne2,A0
                               * Pointer
     move.1
                                * Est-ce l
     cmp.b
              #$0B,D0
               Val Colonne2
     beq
                                    oui al
     move.1
                #Colonne3,A0
                                       er sı
                                                 olonne 3
     bra
               Val_Colonne3
    Gestion de la position d'affich
Val_Colonne0
     move.b
                $10 (A0, D2)
                                       Position de la touche
     clr.b
                D1
     bsr
                     btoxy
                   (D2)
     move.b
     rts
Val Colonnel
                                      * Position de la touche
     move
                   (A0)
     clr.
                 d gotoxy
                0(A0,D2),D0
     rts
Val
     move.b
                $10(A0,D2),D0
                                     * Position de la touche
     clr
                1cd gotoxy
     bsr
     move.b
                0(A0,D2),D0
     rts
Val_Colonne3
     move.b
                                   * Position de la touche
                $10(A0,D2),D0
     clr.b
               D1
     bsr
               lcd_gotoxy
     move.b
               0(A0,D2),D0
     rts
     end
```





EID005_TP 4 DETECTION D'ACTIVATION D'UNE TOUCHE ET LECTURE EN MODE POLLING

4.1 Enoncé du sujet

Objectifs :	Etre capable d'utiliser les utilitaires rangés en bibliothèque, permettant la gestion d'un clavier matricé 16 points (4x4).
Cahier des charges :	Sujet Ecrire un programme en argemble det en C chi réalisera la lecture d'une touche du clay a matric, en prode pooling. Cette lecture se fera par le mode pooling.

Mater al pressaire :

Micro ordinateur de type 7 sous Windows 95 ou ultérieur, Carte mère 16 2 bit à mi ocont Jeur 68332, Réf : EID210000

Carte Clavic Affiche Horlog emps réel : EID00500

Câble de liais US, ou à défaut câble RS232, Réf : EGD000003

Alimer AC, 28V, 1 A Réf : EGD000001,

Documentations nécessaires :

Documents DM Carte Clavier Afficheur Horloge Temps réel : EID005000 Application Notes for the T6963C LCD Graphics Controller Chip (TOSHIBA) T6963c DOT MATRIX LCD CONTROL LSI (TOSHIBA)

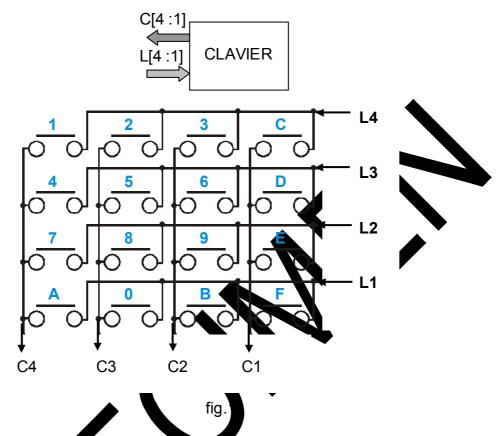
Durée : 1 séance de 3 heures



4.2 Eléments de solution

4.2.1 Description sommaire du clavier

4.2.1.1 Représentation du clavier 4x4 fig.1



Les 4 lignes de l'avier sont cas sur la sortie de l'EPLD de gestion de l'EID005.

Chaque sortice L'EP L'est mur d'une résistance de pull-up (tirage à Vcc).

Les 4 Jonna sont câbées sur des entrées.

4.2.12 Princip général de lecture d'un clavier matricé

Lorsque le sorties commandant les 4 lignes sont au niveau logique 1, les 4 bits correspondant aux 4 colonnes sont à 1 quelque soit le nombre de touches appuyées.

Le principe de la lecture par la méthode pooling consiste à fixer 1 ligne L_j 0, les 3 autres à 1 (balayage lignes) puis à détecter le numéro de la colonne C_i à 0 (balayage colonnes).

L'intersection $L_{j,x}C_i$ donne le caractère correspondant.

Il ne reste plus qu'à coder ce caractère pour lui donner la valeur binaire ou hexadécimale voulue.



4.2.1.3 Détection de l'activation d'une touche appuyée

Cette détection se fait à travers le bit ETAT_CLAVIER (bit 0) du registre STATUS de la carte EID005.

Ce bit STATUS_CLAVIER est positionné à 1 lorsqu'au moins une touche est appuyée.

C'est la variable touche définie dans le fichier de définition eid005.h, ligne suivante :

#define touche status.r bit.b0

4.2.1.4 Organigramme principal

Le clavier est codé par une matrice 4x4 en hexade sin l'con indiqué cidessous.

Attention à la position des ligne de la leau et codage par rapport à celle de la matrice elle-même.

Poids de la colonne	8	4	2	1
	C ₄	C ₃	C ₂	C ₁
1_4	01	02	03	0C
	04	05	06	0D
L2	07	80	09	0E
L1	0A	00	0B	0F

Les lignes et les colonnes sont décrites dans le fichier eid005.h de la carte EID005.



La portion correspondante est ci-dessous.

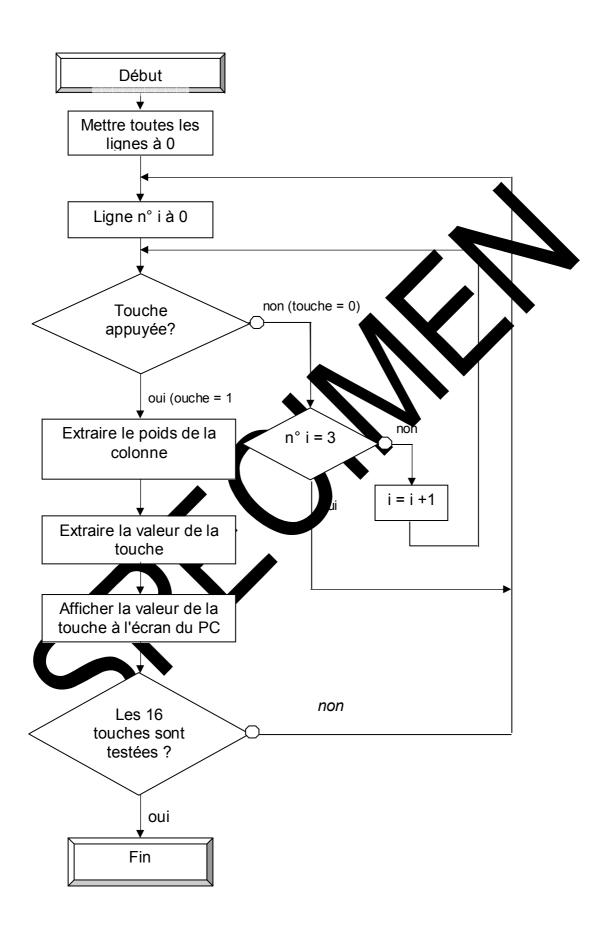
Une ligne i est définie par la variable :

- clavier.matrice.ligne = i ; (0 =< i =< 2 Les colonnes sont lues à travers la variable.
 - > clavier.matrice.colonne.

Important:

Pour utiliser le clavier affi deux avec consilateur C/C++, il faut, dans le menu du logiciel eid210 aller dans le menu configuration puis gnu C/C++. Ensuite aller dans l'onglet tiliner*, cliquer sur *ajouter*, sélectionner le fichier EI 05_Lib.o et finir en cliquant sur ouvrir.







4.2.2 Programme en C

```
/****************
     TP SUR LA CARTE EID005
  **********
    Ecrire un programme en C et qui réalise la
     lecture d'un touche sur un clavier matricé 4 x 4*
    en mode pooling par détection de l'activation d'une touche *
*************
    Nom du FICHIER : EID005 TP4.c
**********************************
    Inclusion des fichiers de définition
#include "eid210.h"
#include "eid005.h"
//===========
// FONCTION PRINCIPALE
//===========
main ()
       Touche [4] [4] = \{ 0x0A, 0x00, 0x0B, 0 \}
short
                      0x07, 0x08, 0x09, 0x
                      0x04, 0x05,
                      0x01, 0x02
short N, V, ValeurTouche;
int tmp;
int i, j, k;
// Préparation pour affichag
                          sur LCD
init aff();
                           ialisati
                                    de l'afficheur
lcd cls();
                             ment
                                   l'afficheur
1cd gotoxy(0x30,00
                                  e la position de départ de la chaîne
de caractères
                      du cl
                            ier en donnant successivement à la
    Balayag
//variable
                  e.ligna les quartets : 1110, 1101, 1011, 0111 .
     clavi
            .matr
print
                ----\n");
      ("TESTER
                TTES LES TOUCHES \n");
prin
                   -----\n\n");
pri
j=0;
do
{j++;
printf ("Appuyer sur une touche \n");
     Touche appuyée ?
    Balayage ligne
while (!touche)
                          // Attente appui touche
     for (i = 0; i<4; i++)
          clavier.matrice.ligne = ~(1 << i); // Toutes les colonnes à 1</pre>
                                         //sauf la nº i
tmp = i ;
```



```
// touche = 1 ==> au moins une touche appuyée
if (touche == 1) break;
for (k = 0 ; k<10000 ; k++);
                               // temporisation
/*****************************
     Extraire le poids de la colonne :
     clavier.matrice.colonne = Elément colonne (4 bits) de la structure
     appartenant à la variable clavier de l'union reg clavier.
     Puis extraire la valeur de la touche appuyée donnée par le poids de
     la colonne :
     Ce poids est donnée par la variable : clavier.matrice.colonne & 0xF
switch((~(clavier.matrice.colonne)) & 0x0F )
     case 1 : ValeurTouche = Touche [tmp][3]; break; //Colonne po
     case 2 : ValeurTouche= Touche [tmp][2]; break; //
     case 4 : ValeurTouche= Touche [tmp][1]; break;
     case 8 : ValeurTouche= Touche [tmp][0]; break
     Afficher loa valeur de la touche à l'ég
printf ("La touche appuyee est : %x\n\n", Val
// Afficher loa valeur de la touche su
                                        <u>é</u>cran
if(ValeurTouche < 0x0A) // Conversion
                                   hex
                                                   chiffres
N= ValeurTouche+0x10;
else
N= ValeurTouche+0x17;
                            version hex
                                            SCII : lettres
                      // E
lcd_write_data(N);
                           oi du caracte.
                                          au LCD
lcd write command(0xC0)
                            // Incrém
                                     tation pour la position du
caractère suivant
while (touche)
                                    Attente relâche touche
                           // Temporisation
for (k = 0 ; k<100)
                              stées ?
                       sont
     Toutes
while (j<1
```



4.2.3Programme en Assembleur

```
TP SUR LA CARTE EID005
     Ecrire un programme en Assembleur
     et en C et qui réalise la lecture d'un touche
     sur un clavier matricé 4 x 4, en mode pooling
   ***********
     Nom du FICHIER : EID005_TP4.src
     *****
     La commande ou la donnée à écrire sont d'aborg
     La donnée lue est rangée dans D0
     La posision x,y d'un caractère est placée
                                               specti
                                                              ns DO et
D1
**
                                                    placée
     L'adresse de début d'une chaîne de car
 *******
     Bits du registre de contrôle de
     ctrlaff :
                b0=rd
                b1=wr
                b2=ce
                b3=cd
                b4=fs
                                       fférents labels
* Inclusion du fichier
                             ant les
* de l'EID210
     include EID2
                   def
     section
     Défini
                    adresses physiques de l'afficheur et du clavier
eid0
                 B30000
                            * Adresse de base Carte clavier_afficheur_rtc
           equ
                   d005+3
                            * Registre contrôle afficheur : bus de
ctr
                            * contrôle
dbaff
                            * Bus de données afficheur
                eid005+4
           equ
                eid005+6
                           * Registre de status de la carte
status
           eq
reg ctrl
                eid005+7
                            * Registre de contrôle de la carte
                EID005+5
reg clavier equ
     Table de définition des paramètres de l'afficheur l'afficheur
TabDef
           dc.b $00,$04,$42
                                  * GH
           dc.b $10,$00,$43
                                  * GA
           dc.b $00,$00,$40
                                  * TH
           dc.b $10,$00,$41
                                  * TA
ModSet
                $80
                                  * OR Graphique ou Texte
           equ
Pointeur
                $24
                                 * Commande pointeur
           equ
DispMod
                $94
                                  * Affichage Texte et ou Graphique
           equ
```



```
AutoInc
                 $C0
                                   * Auto incrémentation pixel ou caractère
           equ
Pile
                                   * Adresse de sauvegarder des contextes
           equ
                 $802000
init_acces equ
                 $EF
                       * Pour accéder au bus de données du lcd avec fs=0
                 $E9
                       * cd=1, ce=0, wr=0, rd=1, fs=0 1110 1001 write
Wrc
           equ
                       * commande
                       * cd=1, ce=0, wr=1, rd=0, fs=0 1110 1010 read
rdc
                 $EA
           equ
                       * commande (Status)
                 $E1
                       * cd=0, ce=0, wr=0, rd=1, fs=0 1110 0001 write data
wrd
           equ
                      * cd=0, ce=0, wr=1, rd=0, fs=0 1110 0010 reda data
rdd
                 $E2
           equ
           dc.b 'EID-----005 $ '
Texte1
           dc.b '?!$'
Texte2
Texte3
           dc.b 'FIN $4
     Tableau des valeurs des touches du clavier
Colonne0
                 dc.b $0C,$0D,$0E,$0F
Colonne1
                 dc.b $03,$06,$09,$0B
Colonne2
                 dc.b $02,$05,$08,$00
Colonne3
                 dc.b $01,$04,$07,$0A
     Tableau des coordonnées d'affichage des
                 dc.b $1A,$3A,$5A,$7A
                 dc.b $18,$38,$58
                 dc.b $16,$36,$56,$
                 dc.b $14,$3
      section
                       code
      PROGRAMME PRING
                                 itialisation Afficheur
     bsr
                                ffacement écran
     bsr
* Envoi Te
                     y=0
                             * LSByte TH : ligne 0 colonne 1
      move.b
                   $00,D0
                 D1
                             * MSByte TH
      bsr
                   ed gotoxy
      mov
                    xte1,A0
                 1cd out str
                   ********
* Définition début affichage touches clavier
      move.1
                 #15,D7
      clr.b
                 D1
```



```
* Lecture du clavier
test clav
    bsr
               Touch
     bsr
              Val_Touch * La valeur de la touche est dans D0
              #$0A,D0 * Conversion Hexa --> ASCII décalé de $20,
     cmp.b
                         * voir Générateur de caractères T6963C :
              sup10
     bcc
              #$10,D0 * add $10 si code < $0A
     add.b
              envoi
    bra
              #$17,D0 * si non add $17
sup10 add.b
envoi bsr
              wr data
     move.b #$C0,D0
     bsr
              wr comm
     dbeq
              D7,test clav
* Envoi Texte2 : x=7, y=0
                          * LSByte TH : ligne 7
     move.b
             #$70,D0
     clr.b
              D1
                          * MSByte TH
               lcd gotoxy
             #Texte2,A0
     move.1
               lcd_out_str
     bsr
* Envoi Texte3 : x=7, y=d
                                                  nne 13
     move.b
               #$7d,D0
                          * LSByte
     clr.b
               D1
                          * MSB
     bsr
               lcd_gotoxy
     move.1
             #Texte3,A0
     bsr
               lcd_out_st;
*==============
     JMP
                                    r au Moniteur
               MONI
                               Reto
*===========
                                    <u>'</u>=========*
     FIN PROGRA
        SO
                 RAMMES
     SOUS
                  GESTION DE L'AFFICHEUR
               #init_acces,ctrlaff
busy move.b
               #rdc,ctrlaff
     move.b
     bsr
               delay
     move.b
               dbaff,D4
                            * Lecture data bus aff
                               * Isoler ST0 STA1 (Status)
     and.b
               #03,D4
               #03,D4
     cmp.b
                               * Si lcd pas prêt
               busy
                               * attendre
     bne
               #init_acces,ctrlaff
     move.b
     rts
```



```
wrcomm : Ecriture commande placée dans D0 ***
**************
wr_comm
     bsr
                busy
                                       * fs = 0; cd, ce, wr et rd = 1
     move.b
                #init_acces,ctrlaff
                D0,dbaff
     move.b
                                       * Déposer la commande sur le bus
                                       * de donnée
                                       * Générer les impulsions
     move.b
                #wrc,ctrlaff
                                       * d'écriture de
                                       * d'une commande
     bsr
                delay
     move.b
                #init acces,ctrlaff
     rts
      wrdata : Ecriture data placée dans D0 ***
wr_data
                busy
     move.b
                #init acces,ctrlaff
     move.b
                D0,dbaff
                                                            le bus de
                                         dc
     move.b
                #wrd,ctrlaff
                                                        .
1sions
                                         d'é
                                                    de
     bsr
                delay
     move.b
                #init_acces,ctrlaff
     rts
     rddata : Lecture da
                           a à placer dan
rd_data
     bsr
                          s,ctrlaff
     move.b
                                       * fs = 0; cd, ce, wr et rd = 1
                                       * Générer les impulsions
     move.b
                                       * d'écriture de
                                       * d'une donnée
     bsr
                  baff,D0
                                       * Lire et mettre la donnée dans D0
     move.b
                #init acces,ctrlaff
```



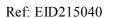
```
init_aff : Initialisation afficheur
init aff
* Initialisation de la zone Adresse de début du Texte
                 D0
     clr.b
                             * LSByte TH = 00
                 wr data
     bsr
                wr_data
                            * MSByte TH = 00
     bsr
                            * Commande d'écriture TH
               #$40,D0
     move.b
                wr comm
                             * Ecriture TH
     bsr
                             * LSByte TA = $10 nombre de cara ères/ligne
                #$10,D0
     move.b
     bsr
                 wr data
                             * MSByte TA = 0
     clr.b
                 D0
     bsr
                 wr data
     move.b
               #$41,D0
                             * Commande d'écriture
     bsr
                 wr comm
                             * Ecriture TA
* Initialisation de la zone Adresse de début du
                                                  aphic
     clr.b
     bsr
                 wr data
                             * LSByte GH = 00
                #04,D0
                             * MSByte \underline{G}H = 00
     move.b
     bsr
                 wr data
                             * Ecritue
                             * Command
     move.b
                #$42,D0
     bsr
                 wr comm
                             * LSByt
     move.b
                 #$10,D0
                                                  re de caractères/ligne
     bsr
                 wr_data
     clr.b
                 D0
                               MSByte GA
     bsr
                 wr_data
     move.b
                 #$43,D0
                               Command
                                        d'écriture TA
                               Ecriture
                                        commande TH
     bsr
                 wr_c
* Définition Modes
                 #Di
     move.b
                          ,D0 * Mse en mode Graphique et/ou Texte
     bsr
                    CC
     move
                    dSei
                              Mise en mode Graphique OU Texte
     bsr
                    comm
     rts
                    Init position caractère
lcd_gotox
                             * Ecrire la donnée LSByte contenue dans D0
     bsr
                 wr_data
     move.b
                 D1,D0
                             * MSByte dans dans D0
                         * Ecrire la donnée MSByte contenue dans DO
     bsr
                 wr data
                 #Pointeur, D0 * Code de commande du Pointeur dans D0
     move.b
     bsr
                 wr comm * Ecriture de la commande du Pointeur dans D0
     rts
```



```
lcd_cls : Effacement écran ***
*********
lcd_cls
                           * Adresse 00,00 dans D0 et D1
     clr.b
                D0
     clr.b
                D1
                lcd gotoxy * Positionner sur le premier caractère
     bsr
               #2048,D3 * Nombre d'octets de la mémoire écran
     move.w
     clr.b
                D0
suite bsr
                wr data
               #AutoInc,D0 * Code auto incrémentation dans D0
     move.b
               wr comm * Incrémenter le pointeur de caractère
     bsr
               #0,D0
     move.b
     sub.w
                #1,D3
                          * Continuer d'effacer tout <u>l'</u>écran
     bne
                suite
     rts
      lcd out str : Envoi chaîne de caractères
                                                   ASCII
** Attention il faut soustraire la valeur $2
                                                            caractère
** lcd_out_str
     move.b
                (A0) + , D0
                            * Pointe
                                         cara
                            * Est_ce
     cmp.b
                #'$',D0
     beq
                            * Fin
                fin
                                     le
                                                 est le $
                            * A cau
     sub.b
                #$20,D0
                                                 I décalé voir notice
     bsr
                wr_data
     move.b
                #AutoInc,D
     bsr
                wr_comm
     bra
                 1cd out st
fin
    rts
      delay
delay move.b
                   10,
bc1
     sub.k
     bne
     rts
     move.1
temp
                  $4000,D5
     sub
tem
                   D5
                 temp1
     rts
```

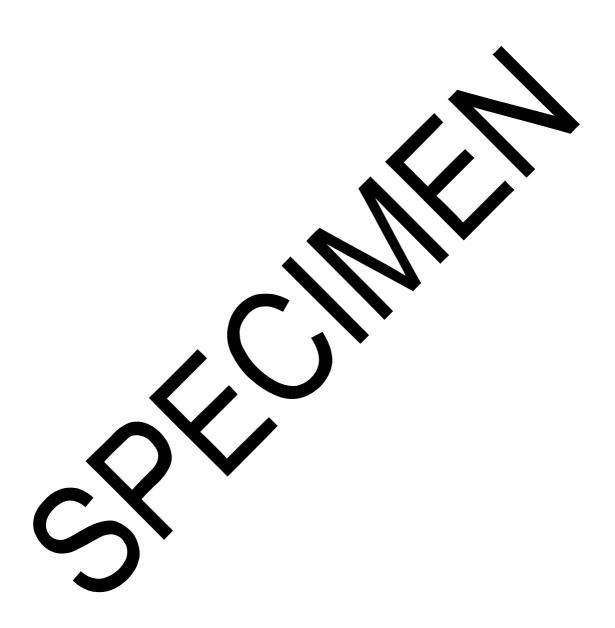


```
***********
     Sous programmes Gestion clavier
*********
    Détection d'activation une touche en mode pooling
     par test du registre status de l'EID005
*****
Touch
               #$EF,D1
                               * Ligne 3
    move.b
               #3,D2
                               * Nº ligne
    move.w
col i move.b
               D1,reg clavier
                              * Pointer une colonne
               reg clavier,D0 * Lire les lignes
    move.b
     and.b
               #$0F,D0
                              * Isoler le quartet colonne
     bsr
               tempo
     btst.b
               #0,status
                               * Test bit 0 du 1
carte : status
                               * Touche appu
                                             pour
                                                        olonne ?
                               * Si aucune
    beq
               balayer
                                            che
                                                           cette
colonne
                               * aller å
    bsr
               Lacher
                                Si oui a
                                                       touche est
relachée
     rts
balayer
     rol.b
               #1,D1
     sub.w
               #1,D2
                                         uivante
                                    ign
    bge
               col i
    bra
               Touch
                                   endre de relâcher la touche
Lacher
     move.b
     bsr
     and.b
                               * Isoler le quartet Colonne
     cmp.b
                               * Touche non relâchée ?
                                Relire les colonnes si touche toujours
     bne
appuyée
                               * Sinon retour de sous prog.
               a valeur de la touche appuyée **
                 ************
Val
     and.1
               #$F,D2
                               * Isoler le n° de la ligne
                              * Pointer sur la colonne 0
     mov
               #Colonne0,A0
     cmp
               #$0E,D0
                              * Est-ce la colonne 0 ?
     beq
                             * Si oui aller lire la valeur
               Val Colonne0
                             * Pointer sur la colonne 1
     move.1
               #Colonne1,A0
                              * Est-ce la colonne 1 ?
     cmp.b
               #$0D,D0
                             * Si oui aller lire la valeur
     beq
               Val_Colonne1
                              * Pointer sur la colonne 2
     move.1
               #Colonne2,A0
                              * Est-ce la colonne 2 ?
     cmp.b
               #$0B,D0
               beq
     move.1
     bra
```





	le la position d'affic	
**********	******	*****
Val_Colonne0 move.b clr.b bsr move.b rts	\$10(A0,D2),D0 D1 lcd_gotoxy 0(A0,D2),D0	* Position de la touche
Val_Colonne1 move.b clr.b bsr move.b rts	\$10(A0,D2),D0 D1 lcd_gotoxy 0(A0,D2),D0	* Position de la touche
Val_Colonne2 move.b clr.b bsr move.b rts	\$10(A0,D2),D0 D1 lcd_gotoxy 0(A0,D2),D0	* Positius de la toutre
Val_Colonne3 move.b clr.b bsr move.b rts	\$10(A0,D2),D0 D1 lcd_gotoxy 0(A0,D2),D0	* sitton de la touche
end		







EID005_TP 5 DESSIN DE LIGNES, CERCLES ET COURBES SUR LE LCD

5.1 Enoncé du sujet

Objectifs :	Etre capable d'utiliser les utilitaires rangés en hibliothèque, permettant le tracé de droites et de cercles sur le fficheur LCD.
Cahier des charges :	Sujet Ecrire un programme en C qui éalise à le tracé de droites verticale, horizontale oblique et à cercu,

Latériel nécessaire :

Micro ordinate. de la le Pasous vindows 95 ou ultérieur, Carte mère (32 bits à micro atrôleur 68332, Réf : EID210000 Carte Clavier à fich ar Horloge Temps réel : EID00500 Câble (32 ison 48, ou à défaut câble RS232, Réf : EGD000003 Alimentation AC/AC 8V, 1 A Réf : EGD000001,

Documentations nécessaires :

Document: DMS Carte Clavier Afficheur Horloge Temps réel: EID005000 Application Notes for the T6963C LCD Graphics Controller Chip (TOSHIBA) T6963c DOT MATRIX LCD CONTROL LSI (TOSHIBA)

Durée : 1 séance de 4 heures



5.2 Eléments de solution

5.2.1 Description de l'afficheur en mode graphique

Attention:

Pour des raisons de conformité avec la documentation du constructeur, les variables x et y représentent respectivement l'ordonnée (axe vertical) et y représente l'abscisse (axe horizontal).

Le point x = 0, y = 0 est en haut à gauche et le point x = 63, y = 127 en bas à droite de l'écran du LCD

Le contrôleur T6963C dispose d'une mémoire de 8 ko.

Mode graphique

La zone graphique est située à partir de l'adresse 2004, se a ajouter à l'octet de poids fort du paramètre GH.

Le LCD en mode graphique contient 64 ligra et 128 join, par agne,. Ce qui donne : 128x64 = 8192 points soit de qui éros allan de 0 à 8191.

Un point est défini en premier lieu par le nu éro de l'octet dans lequel il est situé. Les 8192 points sont donc défine dans 1024 de lets (\$192/8).

Un pixel de cordonnées , y a ur Numé Nu = 128*x + y.

Son adresse Adr da s la médioire ecran (VRAM) est déterminée par le nombre entier d'octets contents ains son numéro Nu ; sa position sera entièrement définie par la minissa se de prinéro du point dans l'octet.

> Adva Part e entière (Nu / 8) ; Adr est sur 2 octets.

Soit = reste de la livision entière de Nu par 8.

Dans soctet, un pixel est repéré par son numéro codé sur 3 bits.

- Le bit le poids faible est allumé à droite de l'octet sur la ligne ; c'est donc pixel dans l'octet et son numéro est 7.
- Le bit de poids fort est allumé à gauche de l'octet sur la ligne ; c'est donc le 1^{er} pixel dans l'octet et son numéro est 0.

Le numéro du point à allumer est donc np = 7- r.

pixel n°7							pixel n°0
np = 0							np = 7
B7	B6	B5	B4	В3	B2	B1	В0



Exemple fig.1

Soit le pixel situé en : x = 18, y = 91

Le numéro du point est Nu = 18x128 + 91 = 2395; Nu / 8 = 299,375

Adr = 299 = 0x012Br = Nu - 299*8 = 4np = 7 - 4 = 3

Ce qui donne les octets du paramètre GH (Graphic Home Adress) :

GH Adress lower = 0x2B GH Adress upper = 0x01 + 0x04.

Les paramètres **GH Adress lower**, **GH Adress upper et np** sont ca**lc**ulés chaque fois qu'un pixel doit être allumé ou éteint.

GRA HIQUE POSITION GRAPHIQUE D'UN POINT DANS LE PLAN EN

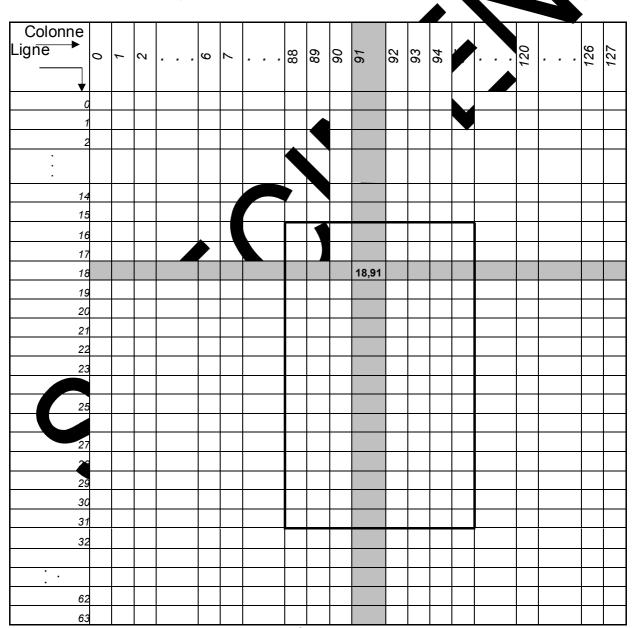


fig.1



5.2.2 Programme principal

Vous disposez de l'utilitaire void Tracer_Pixel (int x, int y, unsigned char Pen) pour tracer des courbes en élaborant votre propre algorithme ou les utilitaires suivants avec leurs commentaires pour aboutir au même résultat.

Les commandes 0xF8 et 0xF0 permettent d'allumer ou d'éteindre le pixel de numéro np.

Cela donne la définition suivante de la variable Pen :

Pen = 0xF8 + np → allumage du pixel de numéro np. Pen = 0xF0 + np → extinction du pixel de • méro np..

Utilitaires Void init_aff() Void lcd_cls() Void lcd_cls() Void lcd_write_command(unsigned char commande) Void lcd_write_data(unsigned char data) Void lcd_clear_TXT() Void lcd_clear_Graph() Void lcd_clear_Graph() Void lcd_gotoxy(unsigned char px, u nswed char py) Void lcd_gotoxy(unsigned char px, u nswed char py) Void lcd_out_standar(xte) Void lcd_out_standar(xte) Void lcd_out_standar(xte) Void lcd_out_standar(xte)		
void lcd_cls() void lcd_write_command(unsigned char commande) void lcd_write_data(unsigned char data) void lcd_write_data(unsigned char data) void lcd_clear_TXT() void lcd_clear_Graph() void lcd_clear_Graph() void lcd_gotoxy(unsigned char px, u nsomed char py) void lcd_out_st_char vxte) lchar vxte	Utilitaires	
void lcd_cls() void lcd_write_command(unsigned char commands) void lcd_write_data(unsigned char data) void lcd_clear_TXT() void lcd_clear_TXT() void lcd_clear_Graph() void lcd_gotoxy(unsigned char px, u nst ned char py) void lcd_out_statchar txxte) void Tracer_ixel(it x, in y, unsigned char Pen) void Tracer_ixel x unatgned char M, unsigned char N, unsigned char P unsigned char P, uns	<pre>void init_aff()</pre>	
void lcd_cls() void lcd_write_command(unsigned char commande) void lcd_write_data(unsigned char data) void lcd_clear_TXT() void lcd_clear_Graph() void lcd_gotoxy(unsigned char px, u nsigned char py) void lcd_out_stribhar(xte) void lcd_out_stribhar(xte) void lcd_out_stribhar(xte) void lcd_out_stribhar(xte) void Tracer_ixel(xtx, invy, unsigned char Pen) void Tracer_LV (unsigned char M, unsigned char Pen) void Droite (int x1, int y1, int x2, int y2, int void Droite (int x1, int y1, int x2, int y2, int void Univer dounce commande void vune dounce pfficer 1'écran texte Ffficer 1'écran Fficure 1'écran texte Ffficer 1'écran Fficure 1'écran Ffficer 1'écran Fficure 1'écran Fficure 1'é		_
void lcd_write_command(unsigned char commande) void lcd_write_data(unsigned char data) void lcd_clear_TXT() void lcd_clear_Graph() void lcd_clear_Graph() void lcd_gotoxy(unsigned char px, u nst ned char py) void lcd_gotoxy(unsigned char px, u nst ned char py) void lcd_out_stribar(xxte) void lcd_out_stribar(xxte) void lcd_out_stribar(xxte) void Tracer_Nixel(xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx		GA, Mode
void lcd_write_data(unsigned char data) void lcd_clear_TXT() void lcd_clear_Graph() void lcd_gotoxy(unsigned char px, u nstrued void lcd_gotoxy(unsigned char px, u nstrued char py) void lcd_out_struckar(xte) void lcd_out_struckar(xte) void Tracer_ixel(xt, in y, unsigned char Pen) void Tracer_ixel(xt, in y, unsigned char Pen) void Tracer_ixel(xt, in y, unsigned char N, unsigned char Pen unsigned char P, unsigned char Q, unsigned char N, unsigned char P, unsigned char Q, unsigned char N, unsigned char P, unsigned char Q, unsigned char Pen idem void Tracer_ixel(xt, in y, unsigned char N, unsigned char P, unsigned char Q, unsigned char N, unsigned char P, unsigned char Q, unsigned char N, unsigned char P, unsigned char Q, unsigned char Pen idem void Tracer_ixel(xt, in y, unsigned char N, unsigned char P, unsigned char		Ef cement le l'écran
void lcd_write_data(unsigned char data) void lcd_clear_TXT() void lcd_clear_Graph() void lcd_clear_Graph() void lcd_gotoxy(unsigned char px, u nst ned char py) void lcd_gotoxy(unsigned char px, u nst ned position d'un caractère ou d'un pixel; px et py sont les données lower et upper. Remarque : pour un pixel il faut ajouter 0x04 à py. void lcd_out_struhar(xte) void Tracer_ixel/it x, in y, unsigned char Pen) void Tracer_ixel/it x, in y, unsigned char N, unsigned char P unsigned char P unsigned char Q, unsigned char N, unsigned char P, usigned char Q, unsigned char N, unsigned char P, unsigned char Q, unsigned char Pen void Tracer_LV (unsigned char M, unsigned char N, unsigned char P, unsigned char Q, unsigned char Pen idem) void Droite (int x1, int y1, int x2, int y2, int idem	<pre>void lcd_write_command(unsigned char commande)</pre>	riture d'une
void lcd_clear_Graph() void lcd_gotoxy(unsigned char px, u ns ned char py) void lcd_gotoxy(unsigned char px, u ns ned char py) void lcd_gotoxy(unsigned char px, u ns ned char py) void lcd_out_st_char voxte) void lcd_out_st_char voxte) void Tracer_ixel(vr x, in y, unsigned char Pen) void Tracer_ixel voxte char py unsigned char N, unsigned char Pen void Tracer_ixel voxte char py unsigned char N, unsigned char Pen void Tracer_ixel voxte char py unsigned char N, unsigned char Pen void Tracer_ixel voxte char py unsigned char N, unsigned char Pen void Tracer_ixel voxte char py unsigned char pen voxte py sont les données lower et upper. Envoi d'une chaîne de caractères pontée par la variable * texte Pen = 0xF8 → Allumer le pixel Pen = 0xF0 → Eteindre le pixel voxte private char py unsigned char pen lower py sont les données du point de départ point de départ void Tracer_LV (unsigned char M, unsigned char Pen void Tracer_LV (unsigned char M, unsigned char Pen void Droite (int x1, int y1, int x2, int y2, int idem		
void lcd_clear_Graph() void lcd_gotoxy(unsigned char px, u ns ned char py) void lcd_gotoxy(unsigned char px, u ns ned char py) void lcd_gotoxy(unsigned char px, u ns ned char py) void lcd_out_st_char voxte) void lcd_out_st_char voxte) void Tracer_ixel(vr x, in y, unsigned char Pen) void Tracer_ixel voxte char py unsigned char N, unsigned char Pen void Tracer_ixel voxte char py unsigned char N, unsigned char Pen void Tracer_ixel voxte char py unsigned char N, unsigned char Pen void Tracer_ixel voxte char py unsigned char N, unsigned char Pen void Tracer_ixel voxte char py unsigned char pen voxte py sont les données lower et upper. Envoi d'une chaîne de caractères pontée par la variable * texte Pen = 0xF8 → Allumer le pixel Pen = 0xF0 → Eteindre le pixel voxte private char py unsigned char pen lower py sont les données du point de départ point de départ void Tracer_LV (unsigned char M, unsigned char Pen void Tracer_LV (unsigned char M, unsigned char Pen void Droite (int x1, int y1, int x2, int y2, int idem	<pre>void lcd_write_data(unsigned char data)</pre>	ritr é d'une donnée
void lcd_gotoxy(unsigned char px, u nsomed position d'un caractère ou d'un pixel; px et py sont les données lower et upper. Remarque : pour un pixel il faut ajouter 0x04 à py. void lcd_out_st_char_exte) Finvoi d'une chaîne de caractères pontée par la variable * texte void Tracer_extel(** t x, in y, unsigned char Pen) Pen = 0xF8 → Allumer le pixel Pen = 0xF0 → Eteindre le pixel void Tracer_LX un igned char Q, unsigned char Pen void Tracer_LV (unsigned char M, unsigned char Pen point de départ P,Q: coordonnées du point de départ P,Q: coordonnées du point de départ	<pre>void lcd_clear_TXT()</pre>	Ef er l'écran texte
void lcd_gotoxy(unsigned char px, u nswned char py) Définition de la position d'un caractère ou d'un pixel; px et py sont les données lower et upper. Remarque: pour un pixel il faut ajouter 0x04 à py. void lcd_out_st_char (xte) Finvoi d'une chaîne de caractères pontée par la variable * texte void Tracer_xixel(xt x, in y, unsigned char Pen) Pen = 0xF8 → Allumer le pixel Pen = 0xF0 → Eteindre le pixel void Tracer_xix y uniqued char M, unsigned char Pen void Tracer_xix y uniqued char Q, unsigned char Pen void Tracer_xix y uniqued char M, unsigned char Pen void Tracer_xix y uniqued char M, unsigned char Pen void Tracer_xix y uniqued char M, unsigned char Pen void Tracer_xix y uniqued char M, unsigned char Pen void Tracer_xix y uniqued char M, unsigned char Pen idem void Droite (int x1, int y1, int x2, int y2, int idem	<pre>void lcd_clear_Graph()</pre>	Effacer l'écran
void lcd_gotoxy(unsigned char px, u nsomed position d'un caractère ou d'un pixel; px et py sont les données lower et upper. Remarque : pour un pixel il faut ajouter 0x04 à py. void lcd_out_stribhar(xxte) Envoid d'une chaîne de caractères pontée par la variable * texte void Tracer_ixel(xt x, in y, unsigned char Pen) Pen = 0xF8 → Allumer le pixel Pen = 0xF0 → Eteindre le pixel void Tracer_Lx unxigned char M, unsigned char N, unsigned char Pen point de départ void Tracer_LV (unsigned char M, unsigned char N, unsigned char P, Q : coordonnées du point de départ void Tracer_LV (unsigned char M, unsigned char N, unsigned char P, unsigned char P, unsigned char Q, unsigned char Pen idem) void Droite (int x1, int y1, int x2, int y2, int idem		aphique
caractère ou d'un pixel ; px et py sont les données lower et upper. Remarque : pour un pixel il faut ajouter 0x04 à py. void lcd_out_struchar(xte) void Tracer_mixel(xt x, in y, unsigned char Pen) void Tracer_mixel(xt x, in y, unsigned char Pen) void Tracer_mixel(xt x, in y, unsigned char Pen) void Tracer_mixel(xt x, in y, unsigned char N, unsigned char Pen e) void Tracer_mixel(xt x, in y, unsigned char N, unsigned char Pen e) void Tracer_mixel(xt x, in y, unsigned char N, unsigned char Pen e) void Tracer_mixel(xt x, in y, unsigned char N, unsigned char Pen e) void Tracer_mixel(xt x, in y, unsigned char N, unsigned char Pen e) void Tracer_mixel(xt x, in y, unsigned char N, unsigned char Pen e) void Tracer_mixel(xt x, in y, unsigned char N, unsigned char Pen e) void Tracer_mixel(xt x, in y, unsigned char N, unsigned char Pen e) void Tracer_mixel(xt x, in y, unsigned char N, unsigned char Pen e) void Tracer_mixel(xt x, in y, unsigned char N, unsigned char Pen e) void Tracer_mixel(xt x, in y, unsigned char N, unsigned char Pen e) void Tracer_mixel(xt x, in y, unsigned char N, unsigned char Pen e) void Tracer_mixel(xt x, in y, unsigned char N, unsigned char Pen e) void Tracer_mixel(xt x, in y, unsigned char N, unsigned char Pen e) void Tracer_mixel(xt x, in y, unsigned char Pen e) void Tracer_mixel(xt x, in y, unsigned char N, unsigned char Pen e) void Tracer_mixel(xt x, in y, unsigned char N, unsigned char Pen e) void Tracer_mixel(xt x, in y, unsigned char Pen e) void Tracer_mixel(xt x, in y, unsigned char Pen e) void Tracer_mixel(xt x, in y, unsigned char Pen e) void Tracer_mixel(xt x, in y, unsigned char Pen e) void Tracer_mixel(xt x, in y, unsigned char Pen e) void Tracer_mixel(xt x, in y, unsigned char Pen e) void Tracer_mixel(xt x, in y, unsigned char Pen e) void Tracer_mixel(xt x, in y, unsigned char Pen e) void Tracer_mixel(xt x, in y, unsigned char Pen e) void Tracer_mixel(xt x, in y, unsigned char Pen e) void Tracer_mixel(xt x, in y, unsigned char Pen e) void Tracer_mixel		Définition de la
pixel; px et py sont les données lower et upper. Remarque: pour un pixel il faut ajouter 0x04 à py. Void lcd_out_statchar (xxte) Envoi d'une chaîne de caractères pontée par la variable * texte Pen = 0xF8 → Allumer le pixel Pen = 0xF0 → Eteindre le pixel Void Tracer_Enx (unaigned char M, unsigned char N, unsigned char Punsigned char Q, unsigned char Pen) Void Tracer_LV (unsigned char M, unsigned char N, unsigned char P, unsigned char Q, unsigned char Pen) void Tracer_LV (unsigned char M, unsigned char N, unsigned char P, unsigned char Q, unsigned char Pen) void Droite (int x1, int y1, int x2, int y2, int idem		position d'un
px et py sont les données lower et upper. Remarque: pour un pixel il faut ajouter 0x04 à py. void lcd_out_stathar(xxte) void Tracer_Nixel(xt x, in y, unsigned char Pen) void Tracer_Nixel(xt x, in y, unsigned char N, unsigned char Pen point de départ void Tracer_Nixel(xt x, in y, unsigned char N, unsigned char N, unsigned char Pen point de départ void Tracer_Nixel(xt x, in y, unsigned char N, unsigned char Pen point de départ void Tracer_Nixel(xt x, in y, unsigned char N, unsigned char Pen point de départ void Tracer_Nixel(xt x, in y, unsigned char N, unsigned char Pen point de départ void Tracer_Nixel(xt x, in y, unsigned char N, unsigned char Pen point de départ void Tracer_Nixel(xt x, in y, unsigned char N, unsigned char Pen point de départ void Tracer_Nixel(xt x, in y, unsigned char N, unsigned char Pen point de départ void Tracer_Nixel(xt x, in y, unsigned char N, unsigned char Pen point de départ void Tracer_Nixel(xt x, in y, unsigned char N, unsigned char Pen point de départ void Tracer_Nixel(xt x, in y, unsigned char N, unsigned char Pen point de départ void Tracer_Nixel(xt x, in y, unsigned char N, unsigned cha	char py)	caractère ou d'un
données lower et upper. Remarque : pour un pixel il faut ajouter 0x04 à py. void lcd_out_struhar(xxte) Envoi d'une chaîne de caractères pontée par la variable * texte void Tracer_ixel(xt x, in y, unsigned char Pen) Pen = 0xF8 → Allumer le pixel Pen = 0xF0 → Eteindre le pixel Pen = 0xF0 → Eteindre le pixel void Tracer_ixel(xt x, in y, unsigned char N, unsigned char Pen pixel Pen = 0xF0 → Eteindre le pixel void Tracer_ixel(xt x, in y, unsigned char N, unsigned char Pen pint de départ void Tracer_ixel(xt x, in y, unsigned char N, unsigned char Pen pint de départ void Tracer_ixel(xt x, in y, unsigned char N, unsigned char Pen pint de départ void Tracer_ixel(xt x, in y, unsigned char N, unsigned char N, unsigned char Pen pint de départ void Tracer_ixel(xt x, in y, unsigned char N, unsigned char N, unsigned char Pen pint de départ void Tracer_ixel(xt x, in y, unsigned char N, unsigned char N, unsigned char N, unsigned char Pen pint de départ void Tracer_ixel(xt x, in y, unsigned char N, unsigned char N, unsigned char Pen pint de départ void Tracer_ixel(xt x, in y, unsigned char N, unsigned char N, unsigned char N, unsigned char N, unsigned char Pen pint de départ void Tracer_ixel(xt x, in y, unsigned char N, unsigned		<pre>pixel ;</pre>
upper. Remarque: pour un pixel il faut ajouter 0x04 à py. void lcd_out_strichar(xte) Envoi d'une chaîne de caractères pontée par la variable * texte void Tracer_Nixel(xt x, in y, unsigned char Pen) Pen = 0xF8 → Allumer le pixel Pen = 0xF0 → Eteindre le pixel Pen = 0xF0 → Eteindre le pixel void Tracer_LN unsigned char M, unsigned char N, unsigned char Punsigned char M, unsigned char Pen void Tracer_LV (unsigned char M, unsigned char N, unsigned char P, unsigned char Q, unsigned char Pen void Tracer_LV (unsigned char M, unsigned char N, unsigned char P, unsigned char Q, unsigned char Pen void Tracer_LV (int x1, int y1, int x2, int y2, int idem		px et py sont les
remarque : pour un pixel il faut ajouter 0x04 à py. void lcd_out_stribar (xte) void Tracer_Nixel(xt x, in y, unsigned char Pen) void Tracer_Nixel(xt x, in y, unsigned char N, unsigned char Pen) void Tracer_Nixel(xt x, in y, unsigned char N, unsigned char Pen) void Tracer_Nixel(xt x, in y, unsigned char N, unsigned char Pen) void Tracer_Nixel(xt x, in y, unsigned char N, unsigned char N, unsigned char Pen) void Tracer_LV (unsigned char M, unsigned char N, unsigned char Pen) void Droite (int x1, int y1, int x2, int y2, int idem		données lower et
void lcd_out_stribar (xte) void lcd_out_stribar (xte) void Tracer_ixel(xt x, in y, unsigned char Pen) void Tracer_ixel(xt x, in y, unsigned char N, unsigned char Pen) void Tracer_ixel(xt x, in y, unsigned char N, unsigned char Pen) void Tracer_ixel(xt x, in y, unsigned char N, unsigned char Pen) void Tracer_ixel(xt x, in y, unsigned char N, unsigned char Pen) void Tracer_ixel(xt x, in y, unsigned char N, unsigned char Pen) void Tracer_ixel(xt x, in y, unsigned char N, unsigned char Pen) idem void Tracer_ixel(xt x, in y, unsigned char N, unsigned char Pen) idem		upper.
void lcd_out_str(har vxte) void lcd_out_str(har vxte) Envoi d'une chaîne de caractères pontée par la variable * texte Pen = 0xF8 → Allumer le pixel Pen = 0xF0 → Eteindre le pixel void Tracer_LN unvigned char M, unsigned char N, unsigned char P unsigned char Q, unsigned char Pen void Tracer_LV (unsigned char M, unsigned char N, unsigned char P, unsigned char Q, unsigned char Pen void Tracer_LV (unsigned char M, unsigned char N, unsigned char P, unsigned char Q, unsigned char Pen void Droite (int x1, int y1, int x2, int y2, int idem		
void lcd_out_str(har vxte) Envoi d'une chaîne de caractères pontée par la variable * texte Void Tracer_Nixel(**) t x, in y, unsigned char Pen) Pen = 0xF8 → Allumer le pixel Pen = 0xF0 → Eteindre le pixel Void Tracer_N, unsigned char M, unsigned char N, unsigned char Pen N N: coordonnées du point de départ P,Q: coordonnées du point de départ Void Tracer_LV (unsigned char M, unsigned char N, unsigned char Pen) void Droite (int x1, int y1, int x2, int y2, int idem		pixel il faut ajouter
caractères pontée par la variable * texte void Tracer_Nixel(** t x, in y, unsigned char Pen) Pen = 0xF8 → Allumer le pixel Pen = 0xF0 → Eteindre le pixel void Tracer_Louisigned char M, unsigned char N, unsigned char Pen void Tracer_LV (unsigned char M, unsigned char N, unsigned char P, Q : coordonnées du point de départ void Tracer_LV (unsigned char M, unsigned char N, unsigned char P, unsigned char Q, unsigned char Pen void Droite (int x1, int y1, int x2, int y2, int idem		0x04 à py.
void Tracer_Nixel('st x, in y, unsigned char Pen) Pen = 0xF8 → Allumer le pixel Pen = 0xF0 → Eteindre le pixel void Tracer_N unigned char M, unsigned char N, unsigned char P unsigned char Q, unsigned char Pen void Tracer_LV (unsigned char M, unsigned char N, unsigned char P, unsigned char Q, unsigned char Pen void Tracer_LV (unsigned char M, unsigned char N, unsigned char P, unsigned char Q, unsigned char Pen void Droite (int x1, int y1, int x2, int y2, int idem	<pre>void lcd_out_st (shar)</pre>	
void Tracer_Nixel(**\text{t} x, in \ y, unsigned char Pen) Pen = 0xF8 → Allumer le pixel Pen = 0xF0 → Eteindre le pixel void Tracer_N un igned char M, unsigned char N, unsigned char Pen N, N : coordonnées du point de départ P, Q : coordonnées du point de départ void Tracer_LV (unsigned char M, unsigned char N, unsigned char P, unsigned char Q, unsigned char Pen void Droite (int x1, int y1, int x2, int y2, int idem		caractères pontée par
le pixel Pen = 0xF0 → Eteindre le pixel void TraceIN vinsigned char M, unsigned char N, unsigned char P unsigned char Q, unsigned char Pen) void Tracer_LV (unsigned char M, unsigned char N, unsigned char P, unsigned char Q, unsigned char Pen) void Droite (int x1, int y1, int x2, int y2, int idem		la variable * texte
Pen = 0xF0 → Eteindre le pixel void Tracen vunzigned char M, unsigned char N, unsigned char Pen vunsigned char Q, unsigned char Pen void Tracer_LV (unsigned char M, unsigned char N, unsigned char P, unsigned char Q, unsigned char Pen void Droite (int x1, int y1, int x2, int y2, int idem	<pre>void Tracer_ixel(t x, in y, unsigned char Pen)</pre>	Pen = 0xF8 → Allumer
void Trace_IN (unaigned char M, unsigned char N, unsigned char Pen) void Trace_IN (unaigned char Q, unsigned char Pen) void Tracer_LV (unsigned char M, unsigned char N, unsigned char Pen) void Tracer_LV (unsigned char M, unsigned char N, unsigned char Pen) void Droite (int x1, int y1, int x2, int y2, int idem		le pixel
<pre>void Trace DR (un Igned char M, unsigned char N, unsigned char Pen</pre>		Pen = $0xF0 \rightarrow Eteindre$
unsigned char P unsigned char Q, unsigned char Pen point de départ P,Q: coordonnées du point de départ void Tracer_LV (unsigned char M, unsigned char N, unsigned char P, unsigned char Q, unsigned char Pen) void Droite (int x1, int y1, int x2, int y2, int idem		le pixel
P,Q: coordonnées du point de départ void Tracer_LV (unsigned char M, unsigned char N, unsigned char P, unsigned char Q, unsigned char Pen) void Droite (int x1, int y1, int x2, int y2, int idem	void Trace un igned char M, unsigned char N,	M, N : coordonnées du
void Tracer_LV (unsigned char M, unsigned char N, unsigned char P, unsigned char Q, unsigned char Pen idem void Droite (int x1, int y1, int x2, int y2, int idem	unsigned char Q, unsigned char Pen	point de départ
<pre>void Tracer_LV (unsigned char M, unsigned char N, unsigned char P, unsigned char Q, unsigned char Pen) void Droite (int x1, int y1, int x2, int y2, int idem</pre> idem		
unsigned char P, unsigned char Q, unsigned char Pen) void Droite (int x1, int y1, int x2, int y2, int idem		point de départ
unsigned char P, unsigned char Q, unsigned char Pen) void Droite (int x1, int y1, int x2, int y2, int idem		
<pre>void Droite (int x1, int y1, int x2, int y2, int idem</pre>		
	unsigned char P, unsigned char Q, unsigned char Pen	idem
)	
Pen)	void Droite (int x1, int y1, int x2, int y2, int	idem
	Pen)	





Utilitaires	Commentaires
<pre>void Cercle_NPoints (int x0, int y0, int r,</pre>	x0, y0 : coordonnées
unsigned int Pen, unsigned int N)	du centre
	r : rayon (en nombre
	de points)
	N : nombre de points
	sur le cercle
	Pen = 0xF8 → Allumer
	le pixel
	Pen = 0xF0 → Eteindre
	le pixel

5.2.3 Organigramme du tracé d'une sinusoïde sur un oscina e

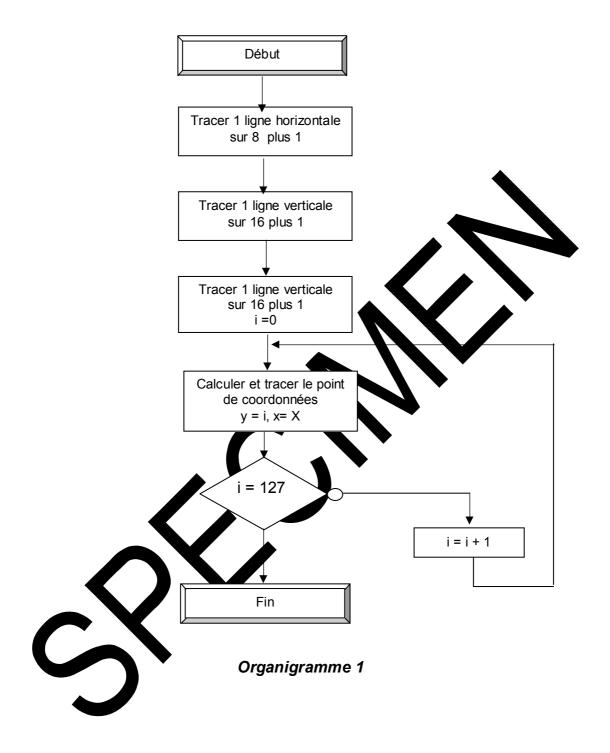
Tracé par point : Organigramme 1

L'angle élémentaire est : pi/N avec (exemple = 3b, Le principe est de calculer X = k*sin (i*pi/N) à la represente an nombre de points, l'amplitude maximale de la sirrusoïde. Le point à tracer a pour cordonnées : Le partir entière de X. abscisse : y ordonnée : x.

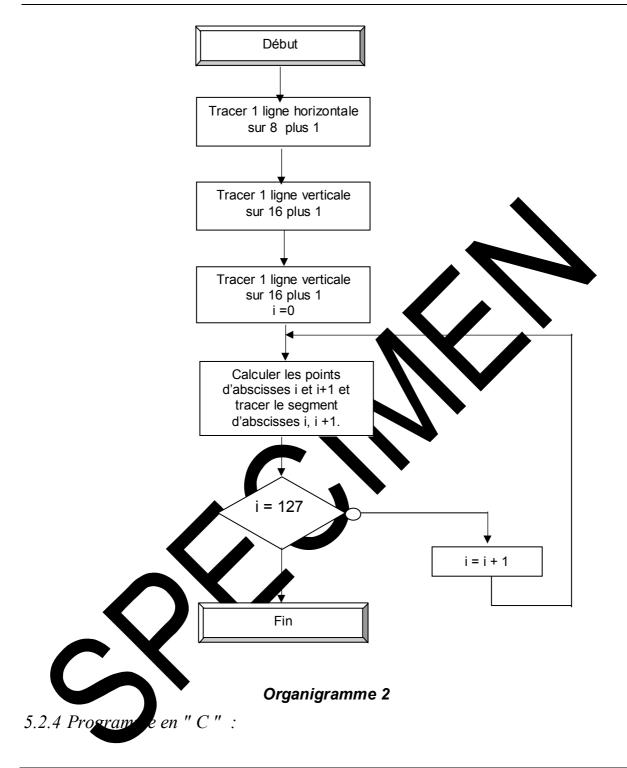
Tracé par segments de d'antes: Organigia ume 2

La méthode est la mêment ceci près qu'il faut calculer les coordonnées de deux points conségnifs et sacer le se ment de droite reliant les deux points.











```
Inclusion des fichiers de définition
#include "eid005.h"
#include <stdio.h>
#include <string.h>
#include <math.h>
//============
// FONCTION PRINCIPALE
//===========
main()
double pi = 3.14159265;
int i, N, m1, m2, k=28; //
                          k : amplitude maximale de
                     Plume baissée : allumer un p
int PB = 0xF8; //
                      Plume levée : éteindre le
int PL = 0xF0;
                //
                //
init_aff();
                      Initialisation de l'a
lcd_cls();
                //
                      Effacement de l'affic
Tab_Sin_Cos ();
                //
                      Calcul sinus et cosinu
                //
                      Pour le tracer
     Exemple : Tracé d'un rectangle
                                                  'afficheur, de ces
                                            horizontale et verticale
     diagonales, et des deux axes de
                                      métr
/********
                                             *********
Tracer LV(0,0,63,0,PB);
Tracer LV(0,127,63,127
Tracer LH(0,0,0,127
Tracer LH(63,0,63,
                   7, PB)
Droite(0,0,63
Droite (63,0,
Cercle NPo
           zs(32,
delay <u>(500</u>000
1cd
                                 // Effacer graphique
               40,
               ,50,25,100,PB);
        15,25,
delay (500000)
lcd clear
```





```
N = 30;
m1 = k*sin(i*(pi/N));
Tracer_Pixel(32-m1,i,PB); // Tracer par POINTS
delay (500000);
lcd clear Graph();
                             // Effacer graphique
// Tracer d'une sinusoïde par SEGMENTS sur un oscilloscope en utilisant la
// void Droite ( int x1, int y1, int x2, int y2, int Pen )
for (i=0; i<64; i+=8)
                                   // Grille 8 lignes verticale
      Tracer_LH(i,0,i,127,PB);
      Tracer_LH(63,0,63,127,PB);
for (i=0; i<127; i+=16)
                                   // Grille 8 lign
      Tracer_LV(0,i,63,i,PB);
      Tracer_LV(0,127,63,127,PB);
for(i=0; i<126; i++)
      m1 = k*sin(i*(pi/30));
      m2 = k*sin((i+1)*(pi/30));
      Droite(32-m1 , i,32-m2, i+1, PB)
      Fin du programme
```



EID005 TP 6 DESSIN D'UNE HORLOGE SUR L'ECRAN GRAPHIQUE

Objectifs :	Etre capable d'utiliser les utilitaires rangés en bibliothèque, permettant la gestion de l'afficheur LCD graphique 128x64 pixels.
Cahier des charges :	Ecrire un programme en assembles et en Coal réalisera le dessin d'une horloge et l'affit par e de l'eur , la date et l'année sur le LCD.

cessaire:

Micro ordinateu sous Wardows 95 ou ultérieur, Carte mère 14 32 bits i mic cor oleur 68332, Réf : EID210000 Carte Clavie, Affiche Horlog, Temps réel : EID00500 Câble de liaiso, U.S., ou à défaut câble RS232, Réf : EGD000003

8V, 1 A Réf : EGD000001, Alim catter AC/

Documentations nécessaires :

Carte Clavier Afficheur Horloge Temps réel : EID005000 Application Notes for the T6963C LCD Graphics Controller Chip (TOSHIBA) T6963c DOT MATRIX LCD CONTROL LSI (TOSHIBA) Real Time Clock DS14285 (DALLAS Semiconductor)

Durée : 1 séance de 4 heures

Ref: EID215040



6.1 Eléments de solution

6.1.1 Définition géométrique de l'horloge

Les 12 repères représentant les 12 heures sont dessinées sur un cercle de rayon R.

Chaque repère est en réalité formé d'un carré dont le centre est sur le cercle de rayon R et de coté égal à 4 pixels.

Le cercle des secondes est identique à celui des heures. Son rayt est de R' et les points sont des carrés de coté 2 pixels, centrés su sercle.

Pour la gestion du LCD, il faut r= se reporter au TP

6.1.1.1 Représentation du circuit

Le modèle du programmeur du circuit DS14285 es crit par la figura

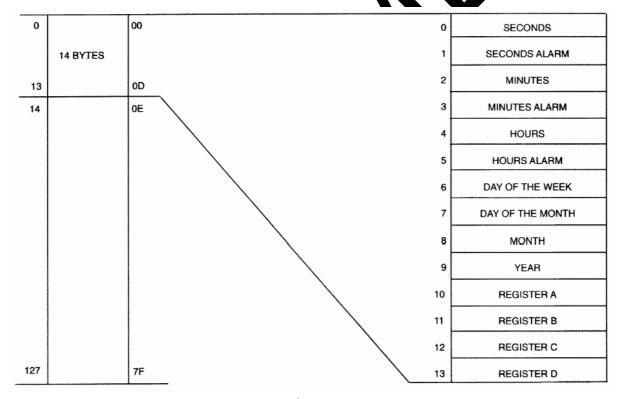


fig.1

Les adresses de 0 à 13 définissent l'emplacement des différents registres du circuit en programmation ou en lecture.



6.1.1.2 Gestion du circuit RTC DS14285

Dans ce TP, seul les deux registres de contrôle A et B du circuit seront programmés.

DS14285/DS14287

CONTROL REGISTERS

The DS14285/DS14287 has four control registers which are accessible at all times, even during the update cycle.

REGISTER A

MSB L						LSB		
	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
	UIP	DV2	DV1	DV0	RS3	RS2	RS1	RS0

UIP - The Update In Progress (UIP) bit is a status flag that can be monitored. When the UIP bit is a 1, the update transfer will soon occur. When UIP is a 0, the update transfer will not occur for at least 244 μs . The time, calendar, and alarm information in RAM is fully available for access when the UIP bit is 0. The UIP bit is read-only and is not affected by RESET . Writing the SET bit in Register B to a 1 inhibits any update transfer and clears the UIP status bit.

DV0, DV1, DV2 - These 3 bits are used to turn the oscillator on or off and to reset the countdown chain. A pattern of 010 is the only combination of bits that will turn the oscillator on and allow the RTC to keep time. A pattern of 11X will enable the oscillator but holds the countdown chain in reset. The next update will occur at 500 ms after a pattern of 010 is written to DV0, DV1, and DV2.

RS3, **RS2**, **RS1**, **RS0** - These four rate-selection bits select one of the 13 taps on the 15-stage divider or disable the divider output. The tap selected can be used to generate an output square wave (SQW pin) and/or a periodic interrupt. The user can do one of the following:

- Enable the interrupt with the PIE bit;
- Enable the SQW output pin with the SQWE bit;
- 3. Enable both at the same time and the same rate; or
- Enable neither.

Table 2 lists the periodic interrupt rates and the square wave frequencies that can be chosen with the RS bits. These 4 read/write bits are not affected by RESET.





DS14285/DS14287

MSB LSB								
	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
	SET	PIE	AIE	UIE	SQWE	DM	24/12	DSE

SET - When the SET bit is a 0, the update transfer functions normally by advancing the counts once per second. When the SET bit is written to a 1, any update transfer is inhibited and the program can initialize the time and calendar bytes without an update occurring in the midst of initializing. Read cycles can be executed in a similar manner. SET is a read/write bit that is not modified by RESET or internal functions of the DS14285/DS14287.

PIE - The periodic interrupt enable PIE bit is a read/write bit which allows the Periodic Interrupt Flag (PF) bit in Register C to drive the IRQ pin low. When the PIE bit is set to 1, periodic interrupts are generated by driving the IRQ pin low at a rate specified by the RS3-RS0 bits of Register A. A 0 in the PIE bit blocks the IRQ output from being driven by a periodic interrupt, but the Periodic Flag (PF) bit is still set at the periodic rate. PIE is not modified by any internal DS14285/DS14287 functions, but is cleared to 0 on RESET.

AIE - The Alarm Interrupt Enable (AIE) bit is a read/write bit which, when set to a 1, permits the Alarm Flag (AF) bit in register C to assert \overline{IRQ} . An alarm interrupt occurs for each second that the 3 time bytes equal the 3 alarm bytes including a "don't care" alarm code of binary 11XXXXXX. When the AIE bit is set to 0, the AF bit does not initiate the \overline{IRQ} signal. The \overline{RESET} pin clears AIE to 0. The internal functions of the DS14285/DS14287 do not affect the AIE bit.

UIE - The Update Ended Interrupt Enable (UIE) bit is a read/write that enables the Update End Flag (UF) bit in Register C to assert IRQ. The RESET pin going low or the SET bit going high clears to UIE bit.

SQWE - When the Square Wave Enable (SQWE) bit is set to a 1, a square wave signal at the frequency set by the rate-selection bits RS3 through RS0 is driven out on a <u>SQW</u> pin. When the SQWE bit is set to 0, the SQW pin is held low; the state of SQWE is cleared by the <u>RESET</u> pin. SQWE is a read/write bit.

DM - The Data Mode (DM) bit indicates whether time and calendar information is in binary or BCD format. The DM bit is set by the program to the appropriate format and can be read as required. This bit is not modified by internal functions or RESET. A one in DM signifies binary data while a 0 in DM specifies Binary Coded Decimal (BCD) data.

24/12 - The 24/12 control bit establishes the format of the hours byte. A 1 indicates the 24-hour mode and a 0 indicates the 12-hour mode. This bit is read/write and is not affected by internal functions of RESET.

DSE - The Daylight Savings Enable (DSE) bit is a read/write bit which enables two special updates when DSE is set to 1. On the first Sunday in April the time increments from 1:59:59 AM to 3:00:00 AM. On the last Sunday in October when the time first reaches 1:59:59 AM it changes to 1:00:00 AM. These special updates do not occur when the DSE bit is a 0. This bit is not affected by internal functions or RESET.



6.1.2Programme principal

Le programme consiste à saisir l'heure, la date, le mois et l'année en cours, puis à vérifier la saisie au fur et à mesure.

Puis il faut formater ces données d'abord pour mettre le circuit RTC à l'heure, l'autoriser à compter le temps et ensuite à afficher les informations à l'écran du LCD,

Enfin il faut lire le RTC et afficher les données toutes les secondes sur le LCD. Le cercle des secondes est effacé au début de chaque minute.

6.1.3 Organigramme

Vous disposez de l'utilitaire void Tracer_Pixel (int x, la y, up gned caur Pen) pour tracer des courbes en élaborant votre propre dornant ou les utilitaires suivants avec leurs commentaires pour aboutir au la ve re utat.

Les commandes 0xF8 et 0xF0 permettent d'umer o d'exindre le pixel de numéro np.

Cela donne la définition suivante de la va jable la va

Pen = 0xE8 + np allugage du pixel de numéro np. Pen = 0xE9 + np axino on du pixel de numéro np.

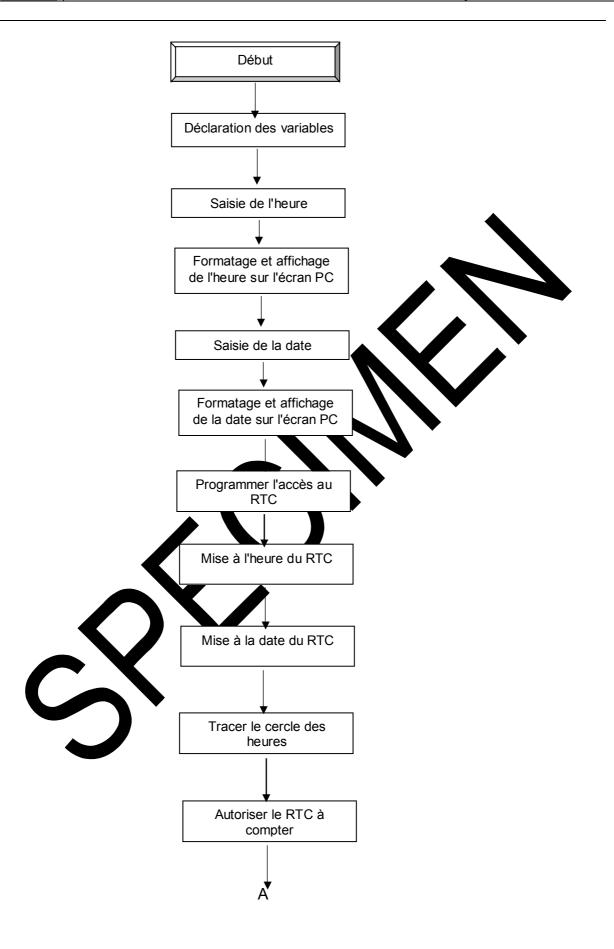
Ut litails	Commentaires
<pre>void init_aff()</pre>	Initialisation des
	paramètres TH, TA, GH,
	GA, Mode
<pre>void lcd_cls()</pre>	Effacement de l'écran
<pre>void lcd_writ _com and _ signe</pre>	Ecriture d'une
	commande
<pre>void lcd_wr e_dat/ unsigne char data)</pre>	Ecriture d'une donnée
void lcd clea T' ()	Effacer l'écran texte
<pre>void a lear_ aph()</pre>	Effacer l'écran
	graphique
	Définition de la
void gotoxy unsigned char px, u nsigned	position d'un
char py)	caractère ou d'un
	pixel ;
	px et py sont les
	données lower et
	upper.
	Remarque : pour un
	pixel il faut ajouter
	0x04 à py.
<pre>void lcd_out_str(char *texte)</pre>	Envoi d'une chaîne de
	caractères pontée par
	la variable * texte
<pre>void Tracer_Pixel(int x, int y, unsigned char Pen)</pre>	Pen = 0xF8 → Allumer
	le pixel
	Pen = $0xF0 \rightarrow Eteindre$
	le pixel



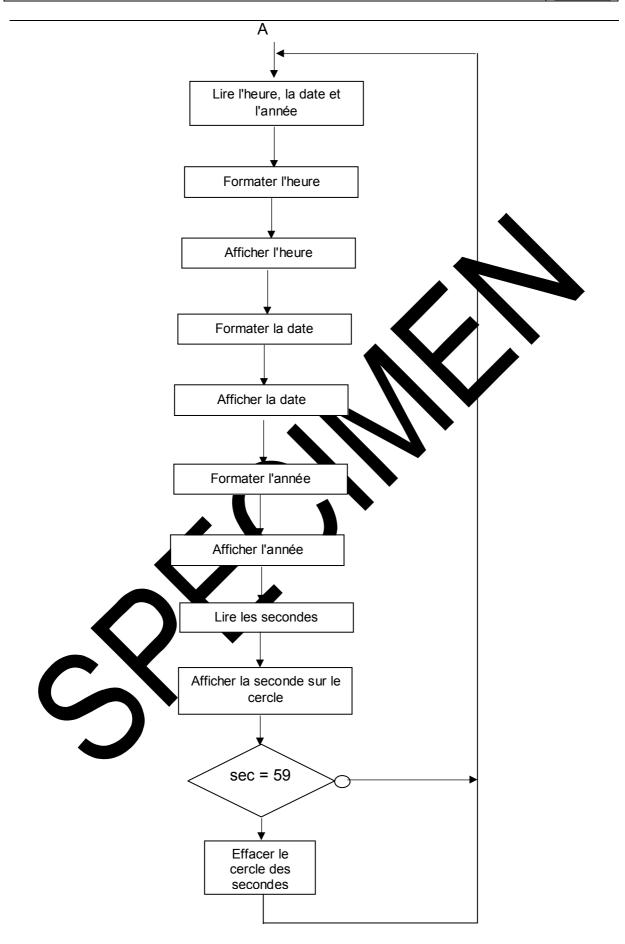
didalab

F	,
<pre>void Tracer_LH (unsigned char M, unsigned char N,</pre>	M, N : coordonnées du
unsigned char P, unsigned char Q, unsigned char Pen	point de départ
	P,Q : coordonnées du
	point de départ
void Tracer_LV (unsigned char M, unsigned char N,	
unsigned char P, unsigned char Q, unsigned char Pen	idem
void Droite (int x1, int y1, int x2, int y2, int	idem
Pen)	
double Cercle_H (int x0, int y0, int r, int Pen,	
int h);	
<pre>void Cercle_S (int x0, int y0, int r, int Pen,</pre>	A
<pre>int sec);</pre>	
void Effacer Cercle S (int x0, int y0, int r, int	
Pen, int sec);	
void writebyte (unsigned char adr, unsigned char	adi
<pre>dat);</pre>	'adr se du tre
	dans le RTC.
unsigned char readbyte(unsigned char adr)	dem











6.1.4 Programme en C

```
/***********************************
     TP SUR LA CARTE EID005
  ***********
     Dessin d'une horloge et l'affichage de l'heure,
     la date et l'année sur le LCD
     Nom du FICHIER : EID005 TP6.c
//
     Nom: Récap OK.c
#include "eid005.h"
#include <stdio.h>
#include <string.h>
#include <math.h>
/*********
/* Variables globales
/**********
double Cos[60], Sin[60]; // Tablea
                                              représentant les
secondes
                                     tion
                                          d'un pixel dans le plan
unsigned char X, Y, np ;
                                       28]
                           xy] = [6
                          np : n° d
                                        k sur la ligne X
                                    pix
double pi = 3.14159265
double deuxpi = 6.283
// K = Facteur de
                                   s dimensions séparant 2 points en x
et en y ;
     K = 1.4872;
* /
     Fonct
                  isées
                          //
    L = 0xF0
                               Plume Levée ==> Effacer
                               Plume Baissée ==> Tracer
main()
           *************
/* DECLARATION DES VARIABLES */
/*********/
char depart, x ;
int R, hr, mn, sc, u, l, k;
int jour, date, mois, an;
int ahr, amn, asc;
char * Jour ;
//char * Jour[7];
char * Mois ;
unsigned char stop ;
     Saisie pour la mise à jour du RTC
```



```
unsigned char Annee[4] = \{ 0x2, 0x0, 0x0, 0x0 \}; // Affichage année
unsigned char DMA [7]; // Saisie Date_Mois_Année unsigned char Date_Mois[5]; // Affichage Date et Mois unsigned char Heure[6]; // Saisie et affichage
unsigned char tmp;
char * Jours[7] = {"Lundi", "Mardi", "Mercr", "Jeudi", "Vendr", "Samed",
"Diman"};
char * Mois[12] = { "Janv", "Fevr", "Mars", "Avri", "Mai", "Juin", "Juill",
"Août",
                 "Sept", "Octo", "Nove", "Decem"};
/**********
/**********
/*
     PROGRAMME PRINCIPAL
                            */
                            */
/**********
/**********
// Calcul sinus cosinus 60 points : secondes
Tab Sin Cos (); // Pour affichage des point
// Initialisation afficheur
init aff();
lcd_cls();
    Saisie de l'heure ou l
                            cer lectur
/* Taper :
           hh = heur
                      puis E
                    tes pu
                            Ent
           mm = m
                          is Entrée.
*/
                        r met
                               e à l'heure \n ou autre pour lancer
printf("
\n");
scanf ("%c
             &depa
if (depart
      printf("
                  Intrer l'heure : hhmmss \n");
     for
                   6; u++)
     scanf("%c
                (x3)
               x08
                 if (u < 0)
                 u = 0;
                 Heure [u] = x \& 0x0F;
                 printf ("
                                         Il reste encore %d
chiffre(s)\n", 5-u);
     else
                                  // Pour effacer les codes de Back Space
                u -= 2;
et Entrée
                printf ("\n RECTIFICATION !\n");
           }
```



```
Mise enforme hhmmss : Tableau Heure[]
Heure[0] = 10*Heure[0] + Heure[1] ; // Heures
     Heure[1] = 10*Heure[2] + Heure[3] ; // Minutes
     Heure[2] = 10*Heure[4] + Heure[5] ; // Secondes
     Verification de l'heure saisie
     printf (" %d h %d mn %d s\t\n", Heure[0], Heure[1], Heure[2]);
/**************************/
    Saisie de la date
/**********
/* Taper :
          j = jour puis Entrée,
          dd = date puis Entrée,
          mm = mois puis Entrée et
          aa = année puis Entrée.
printf (" Entrer la date : jddmmaa \n");
for (u = 0; u < 7; u++)
     scanf("%c",&x);
     if (x != 0x08)
              if (u < 0)
               u = 0;
               DMA[u] = x
               printf("
                           t\t%d \n"
               printf ("
                                       l reste encore %d
chiffre(s)\n", 6-u);
     else
                                 Pour effacer les codes de Back Space
et Entrée
                         RECTIFICATION !\n");
                   *********
               e jddmm : Tableau DMA[]
                 **********
                            // Date
// Mois
// Année
               [1] + DMA [2];
DMA [2] = 10*DM
               [3] + DMA [4];
               [5] + DMA [6];
DMA [3]
     Vérification de la date saisie
printf ("
         %s %d %s 200%d \n", Jours [DMA[0]-1], DMA[1], _Mois [DMA [2]-
1],DMA [3]);
```



```
Programmation du circuit RTC : DS14285
/****************
//--- Mettre à 1 le bit 0 du registre de contrôle du RTC : accès au RTC
ctrl rtc = 1;
//---- Programmation des registres A et B
writebyte (REGB, 0x82); // Registre B = 1000 0010
                     // Autorisation de la mise à jour
                     // Sans incrémentation des compteurs (sec, mm, h).
writebyte (REGA, 0x20); // Registre A = 0010 0000:
//---- Mise à l'heure
writebyte (0, Heure[2]);
writebyte (2, Heure[1]);
writebyte (4, Heure[0]);
/*---- Programmation de l'alarme
writebyte (1, Heure_Alarme [0]);
writebyte (3, Heure_Alarme [1]);
writebyte (5, Heure_Alarme [2]);
//---- Programmation de la Date
                                        semaine : 1 = lun, 2 = mar, ...
writebyte (6, DMA [0]);
                             Jour de
writebyte (7, DMA [1])
                             Date d
                                     s le mois
writebyte (8, DMA [2])
                                     s l'année
                             Mois da
                               nnée
writebyte (9, DMA [3]
                         lancer la RTC
//---- Autoris
                   ion
printf ("
                              e. Appuyer sur RESET pour le stopper \n");
     TRACE
                  ercle heures
for (u = 0; u)
               ); u += 5)
               28, PB, u); // Dessin de l'horloge
Cercle H
Rectangle(0,0,63,127,PB); // Rectangle au tour de l'horloge
//---- Lancement RTC : incrémentation des compteurs (sec, mm, h)
writebyte (REGB, 06);
```



```
/**********************************
    Lecture RTC et affichage de des données
/***********************************
do
{
     do
           tmp = readbyte(REGA);
     while ( tmp & 0x80); // Attente UIP
//----
         Lecture de l'heure
sc = readbyte(0);
mn = readbyte(2);
hr = readbyte(4);
//----
         Lecture jour, date, mois
jour = readbyte(6);
date = readbyte(7);
mois = readbyte(8);
an = readbyte(9);
//----
          Formatage et affichage Heur
                                      minut
     Heure[0] = hr / 10;
     Heure[1] = hr % 10 ;
                           // Code
     Heure[2] = 0x0a;
                                                e l'Afficheur
    Heure[3] = mn / 10 ;
     Heure[4] = mn% 10 ;
     Heure[5] = 0x0a;
                            // idem
     Heure[6] = sc / 10
     Heure[7] = sc % 1
lcd gotoxy(0x55,0x
lcd out Tab(Heure,
                       affic
//----
           25,0x
1cd gotoxy
lcd out_str()
                 our-1]);
                  ge et affichage Date et Mois
               0] = date / 10 ;
                                 // Dizaine Date
     Date Mois
               [] = date % 10 ;
                                 // Unité date
           age du mois en chiffre :
                               // Code ASCII décalé de l'Afficheur -
     Date_Mois[2] = 0xFF ;
0x10
     lcd_gotoxy(0x34,0x0);
lcd_out_Tab(Date_Mois, 2);
lcd_gotoxy(0x37,0x0);
lcd_out_str(_Mois[mois-1]);
```





```
//---- Formatage et Affichage Année
Annee [2] = an / 10 ;  // Dizaine Année
Annee [3] = an % 10 ;  // Unité Année
lcd_gotoxy(0x46,0x0);
lcd_out_Tab(Annee, 4);
//---- Lecture et Affichage des secondes
if (sc == 59)  // Effacer les secondes
{
    Cercle_S (32,63, 28, PB, sc);
    for( u = 0; u<60; u++)
    Effacer_Cercle_S (32,63, 28, PL, u);
}
else
    Cercle_S (32,63, 28, PB, sc);
}
while (1);
}
//---- FIN DU PROGRAMME PRINCIPAL</pre>
```