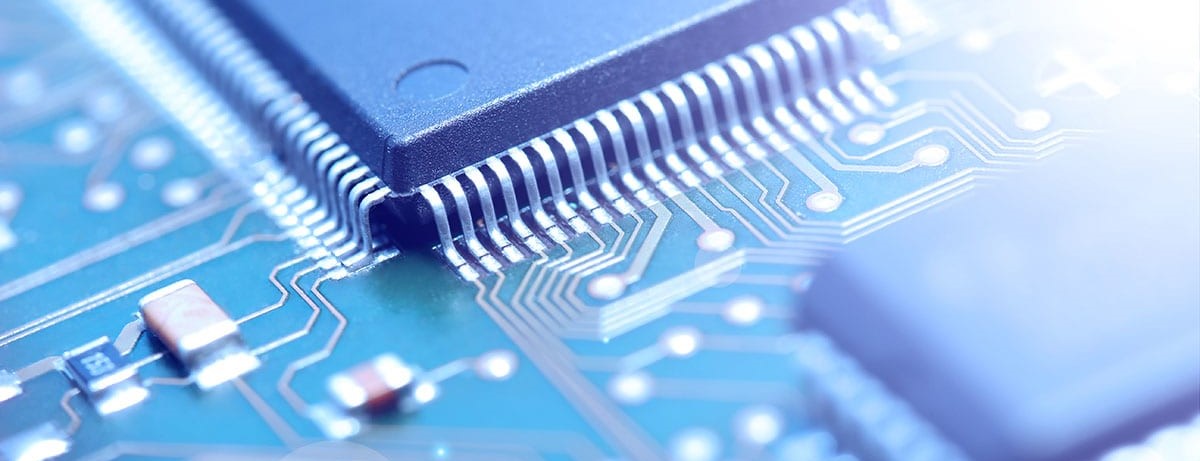


**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**TP MICROCONTROLEUR**

**TP2 – Communication sur un écran LCD et gestion de la mémoire EEPROM**



**Par Damien DUBOIS & Marion ESCOUTELOUP**

**CESI S3E – P1G1 – FIPA26**

**29 juin 2022**

**TABLE DES MATIERES**

[I. INTRODUCTION 3](#_Toc107410645)

[II. Pilotage du clavier et de l’écran LCD 4](#_Toc107410646)

[1. Préparation : récupération des adresses 4](#_Toc107410647)

[2. Ecriture sur l’écran LCD 6](#_Toc107410648)

[3. Utilisation du clavier pour écrire sur l’écran LCD 7](#_Toc107410649)

[III. Développement mémoire EEPROM 8](#_Toc107410650)

[1. Principe et utilisation d’une mémoire EEPROM 8](#_Toc107410651)

[2. Mémorisation de l’affichage 8](#_Toc107410652)

[IV. CONCLUSION 9](#_Toc107410653)

[V. ANNEXE 1 : Programme final 10](#_Toc107410654)

[1. Main.c : fichier source du programme globale 10](#_Toc107410655)

[2. LCD.c : fichier regroupant les fonctions propres à l’écran LCD 12](#_Toc107410656)

[3. Clavier.c : fonctions propres à l’utilisation d’un clavier numérique 13](#_Toc107410657)

[4. Eeprom.c : fonctions pour la mémorisation d’élément dans la mémoire EEPROM 15](#_Toc107410658)

**TABLE DES FIGURES**

[**Figure 1 : Capture d'écran -- Architecture du projet** 3](#_Toc107410659)

[**Figure 2 : Schéma du branchement du multiplexeur 74HCT138** 4](#_Toc107410660)

[**Figure 3 : Tableau description de la datasheet du multiplexeur 74HCT138** 4](#_Toc107410661)

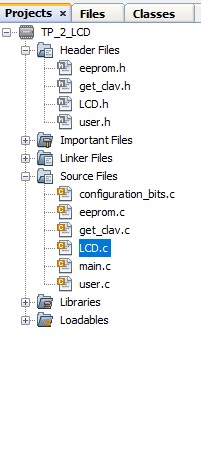
[**Figure 4 : Décryptage des trames pour adressage au clavier et au LCD** 5](#_Toc107410662)

[**Figure 5 : Table de conversion ASCII** 7](#_Toc107410663)

# INTRODUCTION

L’objectif de ce TP2 est de réussir à communiquer avec la mallette d’essai mis à notre disposition. Lors de ce TP, nous nous sommes attardés sur le développement des éléments nécessaire pour le contrôle du clavier ainsi que l’affichage sur l’écran LCD. Dans une seconde partie, nous attarderons sur la mémorisation d’éléments dans une mémoire EEPROM.

Pour réaliser au mieux ce TP, nous nous configurons sur un microcontrôleur de type PIC18F87K22.



**Figure 1 : Capture d'écran -- Architecture du projet**

Pour rappel, nous connaissons les paramètres suivant :

* far unsigned char CLAVIER @ 0x180000 ;
* far unsigned char LCD\_DATA @ 0x1A0002 ;
* far unsigned char LCD\_FUNC @ 0x1A0000 ;

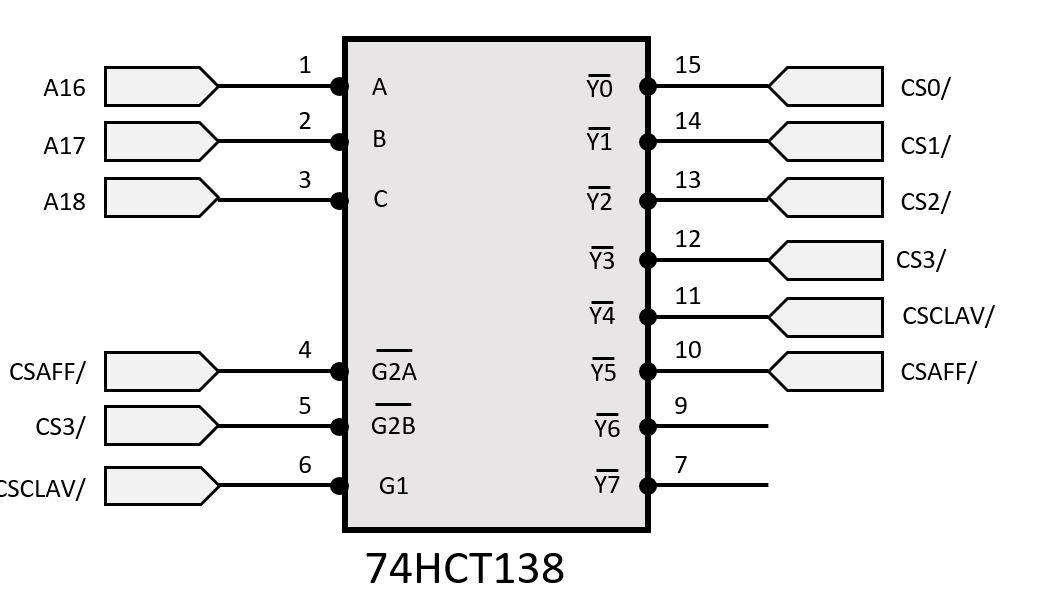
# Pilotage du clavier et de l’écran LCD

## Préparation : récupération des adresses

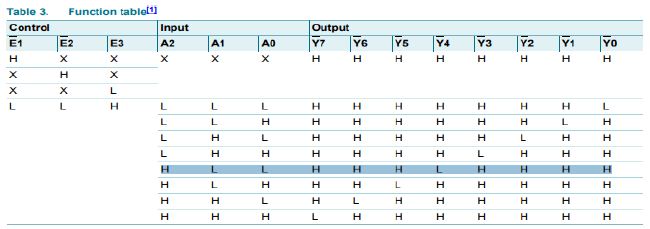
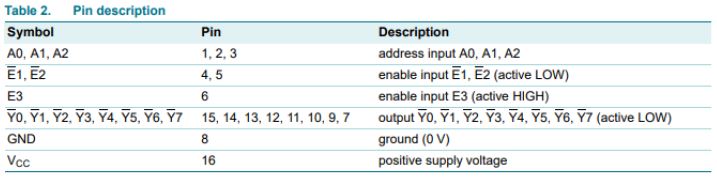
Afin de pouvoir communiquer avec les éléments, il nous est important d'être capable de retrouver leurs adresses. Pour ce faire, dans notre cas, nous pouvons le retrouver grâce à deux documents :

* Le schéma électrique de la mallette
* Datasheet constructeur du multiplexeur

Nous avons ainsi les informations suivantes :



**Figure 2 : Schéma du branchement du multiplexeur 74HCT138**



**Figure 3 : Tableau description de la datasheet du multiplexeur 74HCT138**

Nous pouvons donc, grâce à ces données, retrouver les adressages suivants :

****

**Figure 4 : Décryptage des trames pour adressage au LCD**

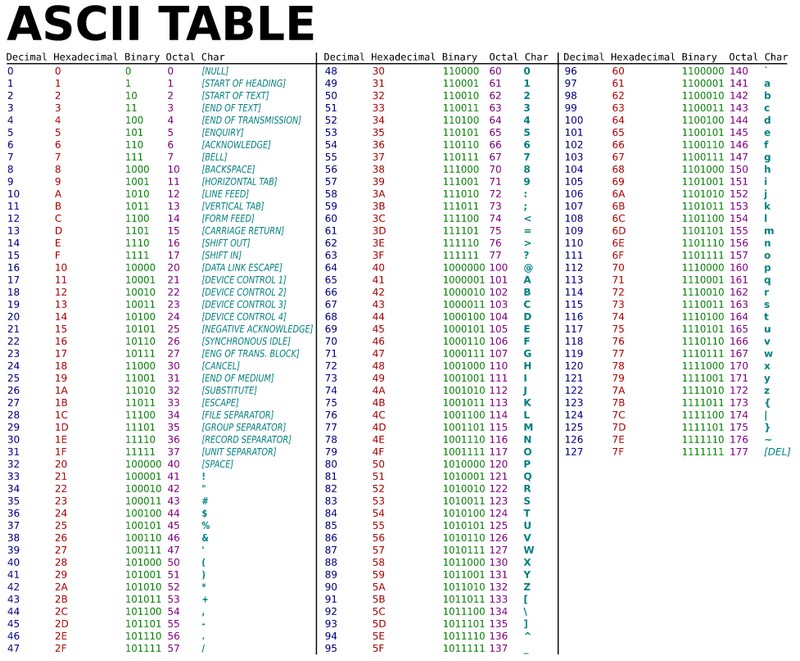
## Ecriture sur l’écran LCD

Maintenant que nous avons l’ensemble des informations, nous pouvons lancer le développement pour pouvoir afficher sur l’écran LCD ce que l’on écrit via le clavier. Pour cela, il ne faut pas oublier d’utiliser les conversions de caractères ASCII. Sans cela, l’affichage ne sera pas juste. Cette conversion sera intégrée directement dans les fonctions propres à l’utilisation du clavier.

Pour pouvoir utiliser correctement l’écran, nous avons besoin de développer deux fonctions. La première, appelé *init\_aff\_lcd()* permet d’initialiser l’écran*.* Il lance toutes les interactions possibles avec cet écran afin de vérifier que l’ensemble fonctionnement bien correctement avant l’application de notre besoin.

Dans un second temps, nous avons programmé la fonction *AffCarac().* Cette fonction, elle, permet de venir directement afficher sur l’écran LCD un ou plusieurs caractères grâce un paramètre d’entrée dans la fonction.

## Utilisation du clavier pour écrire sur l’écran LCD



**Figure 5 : Table de conversion ASCII**

Notre clavier est un clavier numérique, c’est-à-dire qu’il écrit seulement des chiffres allant de 0 à 9. Ainsi, d’après le tableau ASCII ci-dessus, pour afficher ces chiffres sur notre écran LCD nous avons besoin des lignes 0x30 à 0x39.

Utilisation du clavier nécessite l’utilisation d’interruption.

# Développement mémoire EEPROM

## Principe et utilisation d’une mémoire EEPROM

Une mémoire EEPROM est une mémoire dite non-volatile (aucune donnée supprimés lors de la mise hors tension) permettant de stocker des données. Le contenu de cette mémoire peut-être directement changer et modifier de manière quasiment illimité (en réalité modifiable approximativement 1millions de fois au maximum) grâce à un programme. Cette mémoire, interne au microcontrôleur ou externe de part une puce supplémentaire sur un PCB, a plusieurs applications :

* Mémorisation de réglages et configurations pour un système
* Mémorisation de données d’étalonnage
* Mémorisation de mesures sur une longue durée (Data Loger)

Il existe de nombreuses autres applications.

Dans notre cas, nous utilisons la mémoire EEPROM directement présente dans notre microcontrôleur PIC18K87K22. Cette mémoire peut contenir jusqu’à maximum 1024 octets (10 bits).

## Mémorisation en EEPROM

Une image contenant table

Description générée automatiquement

Les deux premiers caractère définissent la case mémoire où écrire et les deux caractères après le « = » définissent la données à sauvegarder dans l’EEPROM.

Nous n’avons pas pu valider cette fonctions de sauvegarde. Après avec fait le code la fonction qui lit la mémoire ne nous renvoi pas la donnée envoyé auparavant dans la même case mémoire. Problème de code ou de mallette, en effet la mallette TP dont nous disposions était quelque peu défaillante.

# CONCLUSION

Ce projet de TP microcontrôleur nous a permis de développer un programme en langage C pour la gestion d’un clavier numérique et d’un affichage sur écran LCD. Indirectement, cela nous a permis d’introduire plusieurs éléments de développement :

* Création de librairies de fonctions (files .c et header .h)
* Gestion des interruptions d’un microcontrôleur
* Gestion d’une conversion ASCII
* Gestion d’une mémoire EEPROM

Malheureusement, nous n’avons tout de même pas réussi à obtenir un résultat satisfaisant sur le développement mémoire de l’EEPROM. Cette partie ne fonctionne pas encore sur la mallette d’essai mise à notre disposition. Plusieurs approfondissements sur ce point sont à prévoir de manière personnels.

# ANNEXE 1 : Programme final

## Main.c : fichier source du programme globale

void main(void) {

////

unsigned char i;

unsigned char addr ;

unsigned char data ;

init();

init\_aff\_lcd(); // init lcd

init\_interrupt();

valeur\_dispo = 0;

j=0;

for(i=0;i>5;i++){

Tab[i]=0;

}

delai\_ms(500);

LCD\_Clear ();

while(1){

if(valeur\_dispo == 1){

decod\_clav();

if(((valeur\_to\_affich >= '0') && (valeur\_to\_affich <= '9')) || (valeur\_to\_affich == '='))

LCD\_DATA = valeur\_to\_affich;

if(valeur\_to\_affich == 1)

LCD\_FUNC = SHIFT\_CURSOR\_LEFT;

if(valeur\_to\_affich == 2)

LCD\_FUNC = SHIFT\_CURSOR\_RIGHT;

if(valeur\_to\_affich == 3){

LCD\_FUNC = DISPLAY\_CLEAR;

while((LCD\_FUNC & 0x80)==0x80);

LCD\_FUNC = RETURN\_HOME;

j=0;

}

if(valeur\_to\_affich == 4){

if(j==6){

j=0;

addr= Tab[0]\*10+Tab[1];

data = (Tab[3]<<4) & Tab[4];

write\_eeprom(addr, data);

Tab[3]=decod\_tab(Tab[3]);

LCD\_DATA = Tab[3];

while((LCD\_FUNC & 0x80)==0x80);

Tab[4]=decod\_tab(Tab[4]);

LCD\_DATA = Tab[4];

}

if(j==3){

j=0;

addr= Tab[0]\*10+Tab[1];

data = read\_eeprom(addr);

LCD\_DATA = data /10;

while((LCD\_FUNC & 0x80)==0x80);

LCD\_DATA = data %10;

}

}

while((LCD\_FUNC & 0x80)==0x80);

valeur\_dispo = 0;

}

}

return;

}

## LCD.c : fichier regroupant les fonctions propres à l’écran LCD

1. void init\_aff\_lcd(void){
2. delai\_ms(100);
3. LCD\_FUNC=0x38; delai\_ms(5);
4. LCD\_FUNC=0x38; delai\_ms(1);
5. LCD\_FUNC=0x38; delai\_ms(1);
6. LCD\_FUNC=SET\_FUNC\_8BIT\_2LINE\_5x7; delai\_ms(1);
7. LCD\_FUNC=SHIFT\_CURSOR\_RIGHT; delai\_ms(1);
8. LCD\_FUNC=DISPLAY\_ON\_CUR\_ON\_BLINKOFF; delai\_ms(1);
9. LCD\_FUNC=ENTRY\_MODE\_SET\_CI\_DNS; delai\_ms(1);
10. LCD\_FUNC=RETURN\_HOME; delai\_ms(2);
11. LCD\_FUNC=DISPLAY\_CLEAR; delai\_ms(2);
12. }

## Clavier.c : fonctions propres à l’utilisation d’un clavier numérique

1. extern unsigned char valeur\_clav;
2. extern unsigned char valeur\_dispo;
3. extern unsigned char valeur\_to\_affich;
4. extern unsigned char j;
5. extern unsigned char Tab[6];
7. // autorisation interruption
8. void init\_interrupt(void)
9. {
10. INTCONbits.GIE = 1;
11. INTCONbits.INT0IE = 1;
12. INTCONbits.INT0IF = 0;
13. }
15. // routine d'interruption
16. void high\_priority interrupt Get\_clav(void)
17. {
18. if(INT0IF)
19. {
20. // valeur de la matrice du clavier
21. valeur\_clav = CLAVIER;
22. Tab[j] = valeur\_clav;
23. valeur\_dispo = 1;
25. j++;
26. // remise a zero du flag
27. INTCONbits.INT0IF = 0;
28. }
30. }
31. void decod\_clav(void)
32. {
33. // selon la valeur de la matrice clavier
34. // valeur à afficher en ascii
35. // non prise en compte du retour arriere ou avancer
36. valeur\_clav = (valeur\_clav & 0x0F);
37. switch(valeur\_clav){
38. case 0 : valeur\_to\_affich = '0'; break;
39. case 1 : valeur\_to\_affich = '1'; break;
40. case 2 : valeur\_to\_affich = '2'; break;
41. case 3 : valeur\_to\_affich = '3'; break;
42. case 4 : valeur\_to\_affich = '4'; break;
43. case 5 : valeur\_to\_affich = '5'; break;
44. case 6 : valeur\_to\_affich = '6'; break;
45. case 7 : valeur\_to\_affich = '7'; break;
46. case 8 : valeur\_to\_affich = '8'; break;
47. case 9 : valeur\_to\_affich = '9'; break;
48. case 11: valeur\_to\_affich = '='; break;
49. case 12: valeur\_to\_affich = 1; break;
50. case 13: valeur\_to\_affich = 2; break;
51. case 14: valeur\_to\_affich = 3; break;
52. case 15 : valeur\_to\_affich = 4; break;
54. default : valeur\_to\_affich = 'x' ;
55. }
56. }
58. unsigned char decod\_tab(unsigned char valeur)
59. {
60. unsigned char tab;
61. // selon la valeur de la matrice clavier
62. // valeur à afficher en ascii
63. // non prise en compte du retour arriere ou avancer
64. valeur = (valeur & 0x0F);
65. switch(valeur)
66. {
67. case 0 : tab = '0'; break;
68. case 1 : tab = '1'; break;
69. case 2 : tab = '2'; break;
70. case 3 : tab = '3'; break;
71. case 4 : tab = '4'; break;
72. case 5 : tab = '5'; break;
73. case 6 : tab = '6'; break;
74. case 7 : tab = '7'; break;
75. case 8 : tab = '8'; break;
76. case 9 : tab = '9'; break;
78. default : valeur\_to\_affich = 'x' ;
79. }
80. return tab;
81. }

## Eeprom.c : fonctions pour la mémorisation d’élément dans la mémoire EEPROM

1. unsigned char read\_eeprom(unsigned char address){
2. int data\_rd;
4. EEADRH = address >> 8;
5. EEADR = address; /\*si un erreur, ici voir EEADRH\*/
7. EECON1bits.EEPGD = 0; /\*Access data EEPROM memory\*/
8. EECON1bits.CFGS = 0; /\*= Access Flash program or data EEPROM memory\*/
9. EECON1bits.RD =1; /\*Initiates an EEPROM read\*/
11. data\_rd = EEDATA; /\*read data\*/
13. return data\_rd;
14. }
16. void write\_eeprom(unsigned char address, unsigned char data\_wr){
17. EEADRH = address >> 8;
18. EEADR = address; /\*si un erreur, ici voir EEADRH\*/
20. EECON1bits.EEPGD = 0; /\*Access data EEPROM memory\*/
21. EECON1bits.CFGS = 0; /\*= Access Flash program or data EEPROM memory\*/
23. EEDATA = data\_wr; /\*ecriture des data\*/
25. EECON1bits.WREN = 1; /\*Allows write cycles to Flash program/data EEPROM\*/
27. INTCONbits.GIE = 0; /\*Disable interrupt\*/
29. EECON2 =0x55; //write sequence unlock
30. EECON2 = 0xAA; //write sequence unlock
32. EECON1bits.WR = 1; //initiates a data EEPROM erase/write cycle
34. while(EECON1bits.WR); //waits for write cycle to complete
36. GIE = 1; //restore interrupts
37. EECON1bits.WREN = 0; //disable write
38. }