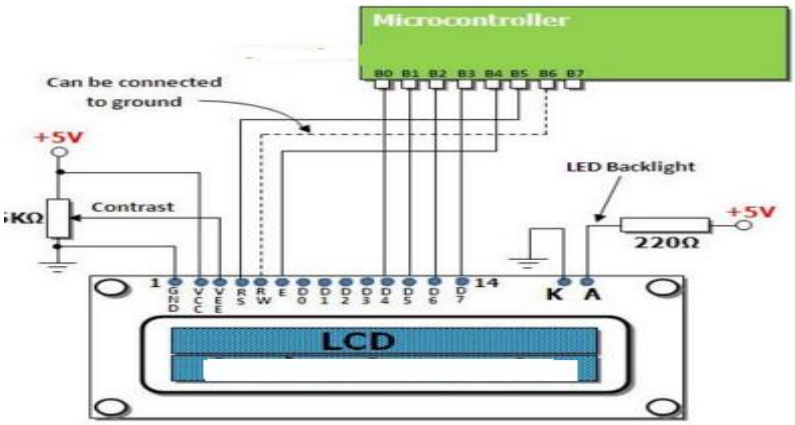
But : **Ecrivez les programmes en MicroC et exécutez sur easypic pour le pic16F877 :**

Soit un LCD a 2 lignes avec le brochage suivant connecté au pic selon la figure ci dessous

1. Ground
2. VCC (+3.3 to +5V)
3. Contrast adjustment (VO)
4. Register Select (RS). RS=0: Command, RS=1: Data
5. Read/Write (R/W). R/W=0: Write, R/W=1: Read
6. Clock (Enable). Falling edge triggered
7. Bit 0 (Not used in 4-bit operation)
8. Bit 1 (Not used in 4-bit operation)
9. Bit 2 (Not used in 4-bit operation)
10. Bit 3 (Not used in 4-bit operation)
11. Bit 4
12. Bit 5
13. Bit 6
14. Bit 7
15. Backlight Anode (+)
16. Backlight Cathode (-)



Un microcontrôleur PIC peut être facilement en communication avec écran LCD en utilisant la bibliothèque de MikroC. l'Interface entre PIC et l'écran LCD peut être de 4 bits ou 8-bits.

Dans le mode 4 bits, la donnée ASCII 8 bits est divisée en 2 parties qui sont envoyées de manière séquentielle à travers les lignes de données DB0 – DB3 vers D4-D7du coté LCD

Définition des connexions LCD

Pour le bon fonctionnement de la bibliothèque LCD, vous devez définir, comment les broches du LCD sont connectés au microcontrôleur PIC

Les définitions ci-dessus indique au compilateur, comment LCD est connecté au microcontrôleur. Les deux ensemble de définitions sont utilisés pour fournir des données (PORT) et Direction (TRIS) registres.

toutes les données transférées à un écran LCD à travers les broches D0-D7 seront interprétées comme une commande ou une donnée, qui dépend de l'état logique de la broches RS: ·RS = 1 - Bits D0 - D7 sont les codes des caractères à afficher. ·RS = 0 - Bits D0 - D7 sont des commandes pour le réglage du mode d'affichage.

```
/**********************************
UN TEXTE A L'ECRAN LCD
=======================
Un contrôleur de texte LCD est relié à
un microcontrôleur PIC dans le mode
(4,5,3,2,1,0). Ce programme affiche les textes
"EMBARQUE" sur la 1 ère ligne
et " Electro Usto " sur la 2ème ligne du LCD.
Oscillateur: HS, 8.0000 Mhz
Compilateur: microC PRO
**********************************/
// Connections de LCD
sbit LCD_RS at RB4_bit;
sbit LCD_EN at RB5_bit;
sbit LCD_D4 at RB0_bit;
sbit LCD_D5 at RB1_bit;
sbit LCD_D6 at RB2_bit;
sbit LCD_D7 at RB3_bit;
sbit LCD_RS_Direction at TRISB4_bit;
sbit LCD_EN_Direction at TRISB5_bit;
sbit LCD_D4_Direction at TRISB0_bit;
sbit LCD_D5_Direction at TRISB1_bit;
sbit LCD_D6_Direction at TRISB2_bit;
sbit LCD_D7_Direction at TRISB3_bit;
// Fin de connections
void main()
{
TRISB = 0;
PORTB = 0xFF;
//TRISB = 0xff;
/* Configurer E/S du portB comme numériques*/
Lcd_Init();                    // Initialiser LCD
Lcd_Cmd(_LCD_CLEAR);
// Effacer un texte sur l'ecrantLCD
Lcd_Cmd(_LCD_CURSOR_OFF);
// Curseur est en off
Lcd_Out(1,2,"EMBARQUE");
 // Ecrire le texte sur la 1 ère ligne
Lcd_Out(2,1,"Electro Usto");
// Ecrire le texte 2 ème ligne}
```

```
// symbole spécial
const char character[] ={0,10,31,31,14,4,0,0};
 const char character2[] ={0,4,17,17,17,31,0,0};

void CustomChar(char pos_row, char pos_char)
{
 char  i;
   LCD_Cmd(64);
   for (i = 0; i<=7; i++)
       LCD_Chr_Cp(character[i]);
   LCD_Cmd(_LCD_RETURN_HOME);
   LCD_Chr(pos_row, pos_char, 0);}
 void main(){
  Lcd_Init();
  Lcd_Cmd(_LCD_CLEAR);
  Lcd_Cmd(_LCD_CURSOR_OFF);
  LCD_Chr(1,4,'I');  //ecrire  I
  CustomChar(1,6);  //dessin du symbole spécial
  Lcd_Out(2,1,"Electronics");
  Delay_ms(3000);
 }
```

Le langage mikroC pour PIC a trouvé une large application pour le développement de systèmes embarqués sur la base de microcontrôleur. Il assure une combinaison de l'environnement de programmation avancée IDE (Integrated Development Environment) , et d'un vaste ensemble de bibliothèques pour le matériel. Le mikroC PRO pour PIC organise des applications dans des projets, composé d'un seul fichier de projet (extension. mcppi) et un ou plusieurs fichiers sources (extension). Le fichier source peut être compilé que si il fait partie d'un projet. La meilleure façon de créer un projet c'est à l'aide de l'Assistant Nouveau projet (menu Project> New ProjeCt)
Sélectionnez le périphérique dans le périphérique dans la liste déroulante.
Saisir la valeur de fréquence de l'oscillateur. Spécifiez l'emplacement où votre projet sera enregistré.

Règles générales d'écriture en microC

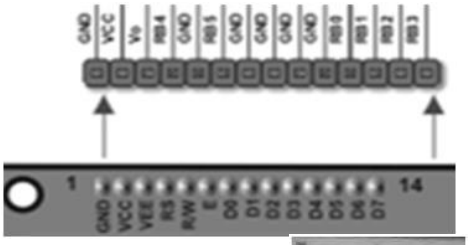
Les instructions propres au langage microC doivent être écrites en minuscule(void main (void)).
Les instructions particulières aux microcontrôleurs doivent être écrites en majuscule (TRISB).
Les retours à la ligne et les espaces servent uniquement à aérer le code
Toutes instructions ou actions se terminent par un point virgule « ; »
Le mikroC LCD bibliothèque fournit un grand nombre de fonctions pour contrôler du texte LCD
Lcd_Config Lcd_Init Lcd_Out Lcd_Out_Cp Lcd_Chr Lcd_Chr_Cp Lcd_Cmd

```
Char  i;
void Move_Delay() {
  Delay_ms(500);  }
void main(){
  Lcd_Init();
  Lcd_Cmd(_LCD_CLEAR);
  Lcd_Cmd(_LCD_CURSOR_OFF);
  Lcd_Out(1,6,txt3);
  Lcd_Out(2,6,txt4);
  Delay_ms(2000);
  Lcd_Cmd(_LCD_CLEAR);
  Lcd_Out(1,1,txt1);
  Lcd_Out(2,5,txt2);
  Delay_ms(2000);
  // deplacement du  text
  for(i=0; i<4; i++) {
    Lcd_Cmd(_LCD_SHIFT_RIGHT);
    Move_Delay();
  }
  while(1) {   // Boucle infinie
    for(i=0; i<8; i++) {
      Lcd_Cmd(_LCD_SHIFT_LEFT);
      Move_Delay();
    }
    for(i=0; i<8; i++) {
      Lcd_Cmd(_LCD_SHIFT_RIGHT);
      Move_Delay();
  }
}}
```

Commande LCD	Description
LCD_CLEAR	Effacer l'affichage
LCD_RETURN_HOME	Retourner vers la position du curseur
LCD_FIRST_ROW	Déplacer le curseur vers la première ligne
LCD_SECOND_ROW	Déplacer le curseur vers la deuxième ligne
LCD_THIRD_ROW	Déplacer le curseur vers la troisième ligne
LCD_FOURTH_ROW	Déplacer le curseur vers la quatrième ligne
LCD_BLINK_CURSOR_ON	Clignotement du curseur
LCD_MOVE_CURSOR_LEFT	Déplacer le curseur à gauche
LCD_MOVE_CURSOR_RIGHT	Déplacer le curseur à droite
LCD_SHIFT_LEFT	Décaler l'affichage à gauche
LCD_SHIFT_RIGHT	Décaler l'affichage à droite

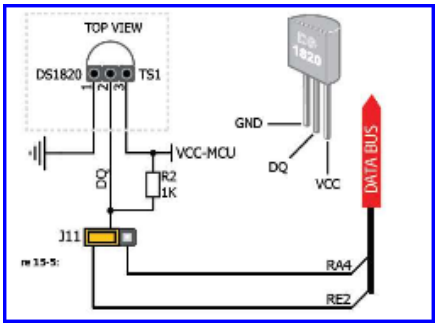
Carte

LCD



Manipulation : Faites la connexion du lcd avec la carte (carte eteinte)
Executer les programmes en exerçant toutes les fonctions possibles
Faites une recherche sur la bibliotheque LCD en MicroC
le deuxieme programme affiche le charactere special ♡ dites comment peut on programmer les lettres en arabe dites comment afficher l'ecran suivant
le troisieme programme affiche 4 textes donner la declaration en initialisant txt? et executer le programme.
Donner le programme complet du convertisseur CAN et le LCD

2ieme partie : **Mettre en œuvre le capteur de Temperature DS1820 et afficher la valeur sur LCD**



Le capteur DS1820 est un capteur numérique qui permet la mesure de la température sur la plage - 55 à +125 °C avec un pas de 0.5 °C. Il se place directement sur la platine de test Easypic5 selon le schéma ci-contre :vérifier que le jumper J11 est placé en position RA5

DS1820 s'appuie sur le protocole one-wire pour communiquer avec le microcontrôleur et transmettre sa mesure. Comme son nom l'indique, un seul fil est nécessaire les commandes pour le faire fonctionner et le contrôler ont été implémentées dans la librairie One_Wire de MikroC. Il y a au total 3 fonctions :

- Ow_Reset permet d'initialiser le capteur – OW_Reset(@du port, n °broche)
- Ow_Read permet la réception des données - OW_Read(@du port, n °broche)
- Ow_Write permet d'envoyer des commandes au capteur – OW_Write(@ du port, n °broche, code commande)

Le protocole de communication One_Wire permet d'utiliser plusieurs capteurs sur un même fil. Chaque capteur est alors identifié par un code sur 64-bits (ROM code). Etant donné que nous n'utilisons qu'un seul capteur, la procédure est simplifiée Pour lire une température et l'afficher, il faut suivre les étapes suivantes :

a) Envoi de la commande CONVERT_T (code Ox44) au capteur (mesure de la température)
b) Envoi de la commande READ_SCRATCHPAD (code 0xBE) au capteur (placement de la température dans le buffer - ou mémoire tampon - du capteur) c) Lecture du buffer d) Affichage

Remarque : avant de transmettre les commandes CONVERT_T et RE AD_ SCRATCHPAD, il faudra au préalable envoyer au capteur une pulse de reset (OW_Reset) puis la commande SKIP ROM (code 0xCC) pour adresser tous les capteurs sans avoir recours au code ROM d'identification (opération coûteuse).

Décodage de la température : A la lecture, le code renvoyé par le capteur est un code binaire sur 16 bits (les 8 premiers marquent le signe) : Avant d'afficher la température, il est donc indispensable de décoder la suite binaire.

```
// LCD module connections
sbit LCD_RS at RB4_bit;
sbit LCD_EN at RB5_bit;
sbit LCD_D4 at RB0_bit;
sbit LCD_D5 at RB1_bit;
sbit LCD_D6 at RB2_bit;
sbit LCD_D7 at RB3_bit;
sbit LCD_RS_Direction at TRISB4_bit;
sbit LCD_EN_Direction at TRISB5_bit;
sbit LCD_D4_Direction at TRISB0_bit;
sbit LCD_D5_Direction at TRISB1_bit;
sbit LCD_D6_Direction at TRISB2_bit;
sbit LCD_D7_Direction at TRISB3_bit;
// End LCD module connections
// Define Messages
char message0[] = "LCD Initialized";
char message1[] = "Room Temperature";
// String array to store temp value to display
char *tempC = "000.0";
unsigned int temp_whole, temp_fraction,
temp_value;
  // 2-byte => 0 .. 65535
signed int tempinC;
 // 2-byte => -32768 .. 32767
unsigned short C_Neg=0, TempH, TempL;
 // 1-byte => 0 .. 255
void Display_Temperature() {
   // check if temperature is negative
   if (temp_value & 0x8000) {
     C_Neg = 1;
     tempC[0] = '-';
     // Negative are stored in 2's complement
     temp_value = ~temp_value + 1;
     }
```

```
    else C_Neg = 0;
    // Get temp_whole
    temp_whole = temp_value >> 4 ;
 // 12 bit >>4 (fraction)  => 8bit whole
    if (temp_value & 0x0001){     // LSB
is 0.0625C
       temp_fraction = 5;
       }
    else temp_fraction = 0;
    tempinC =
temp_whole*10+temp_fraction;
// convert Temp to characters
 if (!C_Neg) {
     if (tempinC/1000)
// 48 is the deci charcode to  display 0
      tempC[0] = tempinC/1000  + 48;
      else tempC[0] = ' '; // space
   }
 tempC[1] = (tempinC/100)%10 + 48;
// Extract tens digit
  tempC[2] =  (tempinC/10)%10 + 48;
 // Extract ones digit
  // convert temp_fraction to charac
 tempC[4] =  tempinC%10  + 48;        //
Extract tens digit
  // print temperature on LCD
  Lcd_Out(2, 6, tempC);
}
void main() {
  ADCON1 = 0xFF;
 // Configure AN pins as digital I/O
  Lcd_Init();   // Initialize LCD
  Lcd_Cmd(_LCD_CLEAR);
  Lcd_Cmd(_LCD_CURSOR_OFF);
```

```
Lcd_Out(1,1,message0);
  Delay_ms(1000);
  Lcd_Out(1,1,message1);
  // Print degree character " °C"
  Lcd_Chr(2,11,223);
  Lcd_Chr(2,12,'C');

    do {
  //--- perform temperature reading
    Ow_Reset(&PORTA, 5);
  // Onewire reset signal
    Ow_Write(&PORTA, 5, 0xCC);
// Issue command SKIP_ROM
    Ow_Write(&PORTA, 5, 0x44);
// Issue command CONVERT_T
    Delay_ms(600);
    Ow_Reset(&PORTA, 5);
    Ow_Write(&PORTA, 5, 0xCC);
// Issue command SKIP_ROM
    Ow_Write(&PORTA, 5, 0xBE);
 // Issue command READ_SCRATCHPAD
    // Read Byte 0 from Scratchpad
    TempL =  Ow_Read(&PORTA, 5);
 // Read => low byte of 2-byte temp registe
    // Then read Byte 1 from Scratchpad
    TempH = Ow_Read(&PORTA, 5);
 // Read => high byte
    temp_value = (TempH << 8)+ TempL ;
 // forming 2-byte value
//--- Format and display result on Lcd
    Display_Temperature();
  } while(1);}
```